Formalizing Dangerous SAT Encodings

Alexander Hertel, Philipp Hertel, & Alasdair Urquhart [*]
Department of Computer Science
University of Toronto

February 4, 2008

Abstract

In this paper we prove an exponential separation between two very similar and natural SAT encodings for the same problem, thereby showing that researchers must be careful when designing encodings, lest they accidentally introduce complexity into the problem being studied. This result provides a formal explanation for empirical results showing that the encoding of a problem can dramatically affect its practical solvability.

We also introduce a domain-independent framework for reasoning about the complexity added to SAT instances by their encodings. This includes the observation that while some encodings may add complexity, other encodings can actually make problems easier to solve by adding clauses which would otherwise be difficult to derive within a Resolution-based SAT-solver. Such encodings can be used as polytime preprocessing to speed up SAT algorithms.

1 Introduction

Satisfiability, or SAT, is the archetypal $\mathcal{NP}$-Complete problem. It has long been known that every problem in $\mathcal{NP}$ can be reduced to SAT using Cook's Theorem [Coo71]. Since propositional formulas are very expressive, instances of many problems in $\mathcal{NP}$ can also be encoded as SAT instances in a much more direct and intuitive way than via Cook's Theorem. This has allowed research into solving these varied problems to be concentrated on SAT-solving. In fact, many problems have numerous different SAT encodings to choose from. Building a framework for comparing the effectiveness of competing encodings is the main focus of this paper.

The strategy of translating problems from other domains to SAT has proved to be fruitful. Nevertheless, this technique is not without its dangers. Empirical evidence suggests that natural encodings which seem to conserve much of the structure of the original problem can actually convert simple instances of the original problem to very difficult SAT formulas. For example, in [KMS96] numerous approaches for translating planning problems to SAT are investigated. Some are found to result in formulas which are much harder to solve than others. This suggests that a great deal of care must be taken in designing encodings since one cannot assume that they will conserve the simplicity of input instances.

This danger is so well-known to the Propositional Reasoning community that the authors of [KMS97] list understanding it as one of ten important and challenging open problems in the area. A more recent follow-up paper [KS03] reaffirms this problem's importance and notes that although some progress has been made, there is still much more work to be done.

We address this problem in two ways. Firstly, we provide a formal example of a natural encoding which translates a trivial instance of the Hamiltonian Cycle problem to an intractably difficult one for any Resolution-based SAT-solver. We also show that very minor modifications to this encoding can make it produce easy SAT instances, thereby formally proving an exponential separation between two very

[*]This research supported by NSERC and the University of Toronto Department of Computer Science

similar and natural encodings. Secondly, we provide a domain-independent framework for comparing the effectiveness of competing encodings, in terms of how easy it is to solve their outputs. This framework relies on the existence of a proof-system hierarchy and the close relationship between SAT-solving and propositional proof complexity.

This paper is organized as follows: In Sections 2 - 4, we provide examples of two very similar and natural SAT encodings for the $\mathcal{NP}$-Complete Hamiltonian Cycle problem, and use them to encode a family of trivially non-Hamiltonian graphs. These encodings are not pathologically designed to create problems, but rather are very intuitive and straightforward. The result of the first encoding is the family of formulas from Theorem 3.1, which we show has exponential lower bounds for AC^0-Frege proof systems, which immediately imply exponential lower bounds for Resolution (RES), and all RES-based SAT-solvers, including Clause Learning.

The result of the second encoding is the family of formulas from Theorem 4.1, which we show to have polynomial upper bounds for Tree Resolution (T-RES) / DPLL, one of the weakest Resolution-based proof systems. This shows that the results of these encodings are respectively very hard and very easy to solve in practice, even though they are very similar and came from the same graph. This result is relevant to the eighth open problem in [KMS97, KS03] because as already mentioned, Propositional Reasoning researchers are aware of empirically tested instances where different reductions can have a significant impact on the complexity of a problem, but this is the first formal example.

In addition, researchers have noted empirically that adding redundant clauses to formulas can transform very difficult instances of SAT into very easy ones. We can further weaken the easy formulas from Theorem 4.1 to obtain the formulas of Corollary 4.2 which contain the clauses of the hard formula from Theorem 3.1 as a proper subset and still have T-RES refutations of polynomial size. This provides a formal example of hard instances which can be converted to easy instances by the addition of redundant clauses.

In Section 5, we describe a formal domain-independent framework which captures our intuitive notions of what constitute dangerous and safe reductions, and show that not only can reductions increase the complexity of problems, but they can beneficially reduce their complexity by applying polytime reasoning which is unavailable to the target proof system.

For the purposes of this paper, we assume that the reader is familiar with the basics of proof complexity and general complexity theory. We shall use [CK01] as our reference for proof complexity. There is a very close relationship between proof-complexity lower bounds and lower bounds for their corresponding algorithms. For example, DPLL is almost identical to the T-RES proof system. Since RES subsumes T-RES, and since RES has exponential lower bounds for the pigeonhole principle [Hak85], it follows that T-RES and therefore DPLL must also have exponential lower bounds for the pigeonhole formulas. In effect, no RES-based SAT-solver will ever be able to solve pigeonhole formulas in polynomial time. Since almost all SAT-solvers are based on refinements of RES, this gives us immediate lower bounds for algorithms such as DPLL and Clause Learning. We use the framework of proof-complexity to categorize encodings as being harmful, neutral, or beneficial.

2 A SAT Encoding For The Hamiltonian Cycle Problem

Consider the following Hamiltonian Cycle to SAT reduction: Take a graph $G = (V, E)$ and create a formula F which enforces a mapping from V to the positions p_1, p_2, ..., $p_{|V|}$ of a Hamiltonian Cycle H. Intuitively, H can be thought of as a cyclic ordering on the $|V|$ vertices of G, where vertex i can be mapped to the j^{th} position in H if i is adjacent in G to the vertices mapped to positions $j-1$ and $j+1$. F contains variables of the form $m_{i,j}$, each of which is interpreted as meaning that element i from G (the domain) is mapped to position j in the Hamiltonian Cycle H (the range). F partially consists of clauses which enforce a bijection between V and H. A conjunction of the following groups of clauses ensure such a bijection:

Total: $\bigwedge_{i=1}^{|V|}(\bigvee_{j=1}^{|V|} m_{i,j})$ *i.e. Every vertex in V maps to at least one position in H.*

Onto: $\bigwedge_{j=1}^{|V|}(\bigvee_{i=1}^{|V|} m_{i,j})$ *i.e. Every position in H has at least one vertex mapped to it.*

1-1: $\bigwedge_{j=1}^{|V|}\bigwedge_{i_1=1}^{|V|}\bigwedge_{\substack{i_2=1\\ i_1\neq i_2}}^{|V|}(\neg m_{i_1,j}\vee\neg m_{i_2,j})$ *i.e. At most one vertex maps to each position.*

Fn.: $\bigwedge_{i=1}^{|V|}\bigwedge_{j_1=1}^{|V|}\bigwedge_{\substack{j_2=1\\ j_1\neq j_2}}^{|V|}(\neg m_{i,j_1}\vee\neg m_{i,j_2})$ *i.e. Every vertex maps to at most one position.*

To ensure that F is satisfiable if and only if G is Hamiltonian, we need only add the following clauses which place constraints on the bijection corresponding to the structure of G:

Edge: $\bigwedge_{j=1}^{|V|}\bigwedge_{i=1}^{|V|}\bigwedge_{\substack{k:(i,k)\notin E\\ i\neq k}}(\neg m_{i,j}\vee\neg m_{k,(j+1)\bmod |V|})$

Informally, the edge constraint clauses are ensuring that for every non-edge (i,k), if vertex i has been mapped to the j^{th} position in the cycle H, then vertex k cannot be mapped to position $j+1 \pmod{|V|}$. It is not hard to see that the reduction is correct: If G is Hamiltonian, then these edge constraints will not cause a contradiction with the clauses enforcing the bijection, so F will be satisfiable. Likewise, if F is satisfiable, then it means that there is a bijection from V to H which respects the constraints enforced by the edge clauses, so G must be Hamiltonian.

Of course, the total, onto, 1-1, and function clauses are more than enough to ensure a bijection. In fact, the total and 1-1 clauses by themselves are sufficient, as are the onto and function clauses by themselves. This leads us to define some notation. Let $H(G)$ be the formula resulting from the above reduction. To this we add a subscript showing which clause groups were used in its construction. We abbreviate total as T, onto as O, 1-1 as 1, and function as F. For example, if we used clauses from the total and 1-1 groups, then the formula is labeled as $H(G)_{T,1}$. There is no need to specify that edge clauses were used, because each of the encodings requires them.

An Interesting Family of Graphs

Consider the complete graph on n vertices, K_n. Let K_n^* be K_n with the addition of a single degree-0 vertex. We shall apply the reductions from the previous section to graphs from this family. Since each K_n^* is disconnected, it is trivially non-Hamiltonian, which in turn means that every formula $H(K_n^*)$ is unsatisfiable.

We will show that K_n^* is interesting because proofs of $H(K_n^*)$ either have polynomial upper bounds for T-RES (and therefore all stronger systems), or exponential lower bounds for AC^0-Frege (and therefore all weaker systems), depending on which clauses are used in its construction. This provides us with a formal example of two very similar and natural encodings whose outputs have drastically different complexities.

3 Exponential Lower Bounds For $H(K_n^*)_{T,1,F}$

In this section we show that the version of the encoding which uses Total, 1-1, and Function clauses, when applied to K_n^* graphs, results in a formula which no Resolution-based SAT-solver can efficiently solve, even though the K_n^* graphs are trivially non-Hamiltonian. In other words, even though the encoding is very natural, it injects an exponential amount of unwanted complexity into our original problem instance.

Theorem 3.1. *Lengths of* $\mathsf{AC^0}$-Frege *proofs for the unsatisfiability of* $H(K_n^*)_{T,1,F}$ *formulas have* $\Omega(2^{\sqrt[5^d]{n}})$ *lower bounds, where* d *is the depth of the Frege proof, and if there exist size-*N $\mathsf{AC^0}$-Frege *proofs restricted by* $m_{x,n} = 1$ *of* $H(K_n^*)_{T,1,F}$*, then there exist size-*$N + O(n^3)$ *proofs of* $fPHP_{n-2}^n$*.*

Proof: The high-level overview of this proof is as follows: Assume that we have a size-N $\mathsf{AC^0}$-Frege proof of $H(K_n^*)_{T,1,F}$. We show that this proof can be restricted with a specially-chosen truth assignment α to get a new, smaller proof of $H(K_n^*)_{T,1,F}\restriction_\alpha$. After unit propagation, this formula becomes $fPHP_{n-2}^n$ which is already known to have exponential $\mathsf{AC^0}$-Frege lower bounds. For those unfamiliar with $\mathsf{AC^0}$-Frege, it suffices to think of this proof in terms of RES; i.e. any size-N RES proof of $H(K_n^*)_{T,1,F}$ can also be restricted to yield $fPHP_{n-2}^n$. Since the lower bound is proved for $\mathsf{AC^0}$-Frege proof systems, we show how to model the unit propagations using $O(n^3)$ steps of $\mathsf{AC^0}$-Frege reasoning. Therefore, if there exists a sub-exponential $\mathsf{AC^0}$-Frege (resp. RES) proof of $H(K_n^*)_{T,1,F}\restriction_\alpha$, then there exists a sub-exponential $\mathsf{AC^0}$-Frege (resp. RES) proof of $fPHP_{n-2}^n$, which is a contradiction, since $fPHP_{n-2}^n$ has exponential lower bounds [BT88].

The details of the proof are as follows: The restriction that we apply to $H(K_n^*)_{T,1,F}$ is $m_{x,n} = 1$. Intuitively, this will guarantee via the edge clauses that we cannot map any vertex to positions $n-1$ or $n+1$ because x has no edges incident on it. If we interpret the variables as mappings from pigeons to holes, we now have two more pigeons than holes. The restriction $m_{x,n} = 1$ propagates as follows:

- For every function clause of the form $(\neg m_{x,n} \vee \neg m_{x,j})$, since we have set $m_{x,n}$ to 1, we must set all of $m_{x,1}, m_{x,2}, ..., m_{x,n-1}$ as well as $m_{x,n+1}$ to 0.
- For every 1-1 clause of the form $(\neg m_{x,n} \vee \neg m_{i,n})$, propagating $m_{x,n} = 1$ causes us to set all of $m_{1,n}, m_{2,n}, ..., m_{n,n}$ to 0.
- Finally, for every edge clause of the form $(\neg m_{x,n} \vee \neg m_{k,n+1})$ where (x, k) is a non-edge in G, propagating $m_{x,n} = 1$ causes us to set all of $m_{1,n+1}, m_{2,n+1}, ..., m_{n,n+1}$ to 0. Similarly, for each edge clause of the form $(\neg m_{i,n-1} \vee \neg m_{x,n})$, propagating causes us to set all of $m_{1,n-1}, m_{2,n-1}, ..., m_{n,n-1}$ to 0.

The effect of these propagations on the various groups is as follows:

- Total Clauses: The restriction $m_{x,n} = 1$ satisfies the clause $(m_{x,1} \vee m_{x,2} \vee ... \vee m_{x,n} \vee m_{x,n+1})$. Combined with this, the propagations $m_{i,n-1} = 0$, $m_{i,n} = 0$, and $m_{i,n+1} = 0$ for all i causes the total clauses to become:
$$\bigwedge_{i=1}^{n}(\bigvee_{j=1}^{n-2} m_{i,j})$$
- 1-1 Clauses: For each $1 \le i \le n+1, i \ne x$, there is a clause $(\neg m_{x,n} \vee \neg m_{i,n})$. Since every $m_{i,n}$ was set to 0, every 1-1 clause involving any $m_{i,n}$ will be satisfied and eliminated. Due to the edge clause propagations, for every $i \ne x$, every clause involving $m_{i,n-1}$ or $m_{i,n+1}$ will also be eliminated. The 1-1 clauses therefore become:
$$\bigwedge_{j=1}^{n-2}\bigwedge_{i_1=1}^{n}\bigwedge_{\substack{i_2=1\\ i_2\ne i_1}}^{n}(\neg m_{i_1,j} \vee \neg m_{i_2,j})$$
- Function Clauses: For each $1 \le j \le n+1, j \ne n$ there is a clause $(\neg m_{x,n} \vee \neg m_{x,j})$. Since every $m_{x,j}$ was set to 0, every function clause involving any $m_{x,j}$ will be satisfied and eliminated. Due to edge clause propagations, for every $i \ne x$, every clause involving $m_{i,n-1}$ or $m_{i,n+1}$ will also be eliminated. Due to the 1-1 clause propagations, for every $i \ne x$, every clause/ involving $m_{i,n}$ will also be eliminated. The function clauses therefore become:

$$\bigwedge_{i=1}^{n} \bigwedge_{j_1=1}^{n-2} \bigwedge_{\substack{j_2=1 \\ j_2 \neq j_1}}^{n-2} (\neg m_{i,j_1} \vee \neg m_{i,j_2})$$

- Edge Clauses: There are two types of edge clauses, those which contain the literal $\neg m_{x,n}$, and those which contain the literal $\neg m_{x,j}, j \neq n$. Note that this covers all edge clauses because vertex x is involved in every non-edge of K_n^*. Clauses of the first type are satisfied by unit propagation which forces the other literal in each such clause to be set to 1. Those of the second type are satisfied by the $m_{x,j}$ propagations from the function clauses. All edge clauses are therefore eliminated.

These remaining clause groups when simplified by unit propagation are exactly the clauses from $fPHP_{n-2}^n$. In effect, the size-N proof of $H(K_n^*)_{T,1,F} \upharpoonright_{m_{x,n}=True}$ has been turned into a proof of $fPHP_{n-2}^n$. It is not hard to show that AC^0-Frege can perform unit propagations in polynomial size for some polynomial $p(n)$. This turns our size-N proof of $H(K_n^*)_{T,1,F} \upharpoonright_{m_{x,n}=True}$ to a size $N + p(n)$ proof of $fPHP_{n-2}^n$.

Let d be the depth bound imposed on a Frege system. Since AC^0-Frege proof systems are closed under restriction (i.e. restricting a proof yields a smaller proof), and since they have $\Omega(2^{\sqrt[5^d]{n}})$ size lower bounds for $fPHP_{n-2}^n$ formulas [UF96], we may conclude that the $H(K_n^*)_{T,1,F}$ formulas also require proofs of size at least $\Omega(2^{\sqrt[5^d]{n}})$. Specifically, if there exist size-N AC^0-Frege proofs of $H(K_n^*)_{T,1,F} \upharpoonright_{m_{x,n}=True}$, then there exist size-$N + p(n)$ proofs of $fPHP_{n-2}^n$. $\square$

Clearly, this result also holds for all formulas such as $H(K_n^*)_{T,1}$ which are composed of proper subsets of the clauses from $H(K_n^*)_{T,1,F}$, because having fewer initial clauses certainly cannot help to find a shorter proof.

Corollary 3.2. *No SAT algorithm based on* AC^0-Frege *nor any weaker proof system can efficiently solve $H(K_n^*)_{T,1,F}$ formulas (or formulas containing a proper subest of those clauses). This includes* DPLL *as well as Clause-Learning based SAT-solver algorithms.*

Therefore we have shown that the $H(G)_{T,1,F}$ encoding can convert trivial instances of the Hamiltonian Cycle problem to intractable SAT instances.

4 Polynomial Upper Bounds For $H(K_n^*)_{T,O,F}$

In this section we show that the version of the encoding which uses Total, Onto, and Function clauses, when applied to K_n^* graphs, results in a formula which has short DPLL proofs. This is particularly interesting because both $H(K_n^*)_{T,O,F}$ and $H(K_n^*)_{T,1,F}$ are natural encodings of the Hamiltonian Cycle problem, and neither is a subset of the clauses of the other, but $H(K_n^*)_{T,O,F}$ is easy to solve, while $H(K_n^*)_{T,1,F}$ is intractably difficult.

Theorem 4.1. T-RES *proofs for the unsatisfiability of $H(K_n^*)_{T,O,F}$ formulas have $O(n^2)$ size upper bounds, where n is the number of distinct variables contained in the formulas.*

Proof: For the following argument, please refer to the T-RES proof template shown in Figure 1. Note that some of the leaves are labelled with the specific clauses which are falsified at that position. In addition, to avoid diagramatic clutter, the remaining leaves are labelled with the groups containing the clauses which are falsified.

We initially branch on $m_{x,1}$. Since x is an isolated vertex in K_n^*, setting $m_{x,1} = 1$ ensures that assigning any future vertex to position 2 (i.e. setting $m_{i,2} = 1$ for any $i \neq x$) will falsify an edge clause, and therefore falsify the formula. Setting $m_{x,2} = 1$ also falsifies $H(K_n^*)_{T,O,F}$ by falsifying a function clause, because no vertex may be assigned to more than one position. Finally, setting $m_{i,2} = 0$ for $i = 1, 2, ..., n, x$ will falsify the onto clause requiring that some vertex be mapped to position 2. This subtree requires $2n + 3$ nodes.

When we set $m_{x,1} = 0$, we next branch on $m_{x,2}$. Clearly, the formula rooted by setting $m_{x,2} = 1$, can be shown to be unsatisfiable with a tree of size $2n + 3$ for the same reasons as above.

For each i, after setting $m_{x,i-1} = 0$, we branch on $m_{x,i}$. Each positive branching will result in a subtree of size $2n + 3$ as described above.

The all-negative assignment to every $m_{x,i}$ falsifies the total clause ensuring that vertex x is mapped to some position, after which all branching is complete. This T-RES proof therefore has size $(n+1)(2n+3)+1$, which is $O(n^2)$, as required. □

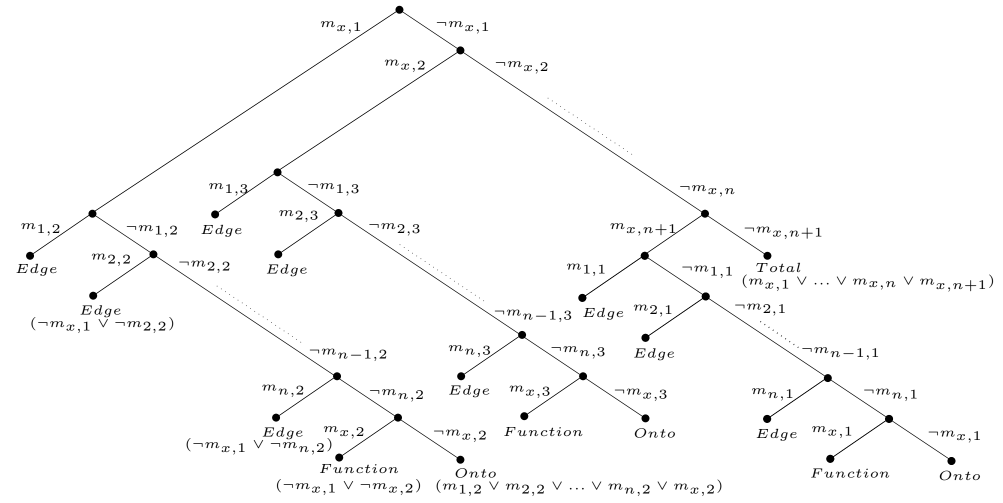

Figure 1: A Template for Polynomially-Sized T-RES Proofs for the Unsatisfiability of $H(K_n^*)_{T,O,F}$ Formulas

Clearly, this result also holds for all formulas such as $H(K_n^*)_{T,O,1,F}$ which are composed of proper supersets of the clauses from $H(K_n^*)_{T,O,F}$, because having more initial clauses certainly cannot hurt when finding short proofs:

Corollary 4.2. *The size of* T-RES *proofs for the unsatisfiability of $H(K_n^*)_{T,O,1,F}$ formulas have polynomial upper bounds.*

Corollary 4.3. *For any $H(K_n^*)_{T,O,F}$ formula (or formula containing a superset of those clauses), there is a polynomially-bounded* DPLL *computation which solves it, assuming that the variables to branch on are chosen in the correct order.*

5 Domain Independent Framework for Comparing Encodings

Currently, no system exists to classify encodings according to whether they make problem instances harder or easier. Such a classification system might prove to be very beneficial for researchers who are actively using SAT-solvers to tackle $\mathcal{NP}$-Complete problems. It might also prove to be beneficial for researchers who are interested in studying the phenomenon of dangerous encodings more abstractly with an eye to finding general principles for predicting which encodings will lead to complexity blow-ups on certain families of formulas. In this section, we provide a framework for such a system.

Although no such system has yet been devised, much work has gone into classifying the power of proof systems. These proof systems have been organized into a hierarchy based on polynomial simulations and exponential separations. Briefly, a proof system α is said to p-simulate another proof system β if for every unsatisfiable CNF formula f, there exists an α refutation of f which is at most a polynomial factor larger than f's smallest β refutation. A proof system α is said to be exponentially separated from another proof system β if there exists some class of formulas F such that for all $f \in F$ there exists an α refutation of f with polynomial size, but the smallest β refutation of f has exponential size. Such a separation clearly implies that β cannot p-simulate α. If α p-simulates β and is also exponentially separated from β, then we say that α is strictly stronger than β, and there is always a (nondeterministically chosen) computation of a SAT-solving algorithm based on the principles of α which will finish within a polynomial factor of the time it would take any SAT-solver based on the principles of β to finish.

Much work has gone into establishing a proof system hierarchy especially for systems based on the Resolution rule. Please refer to Figure 2 below for the portion of the hierarchy which is particularly relevant to propositional reasoning and SAT solving.

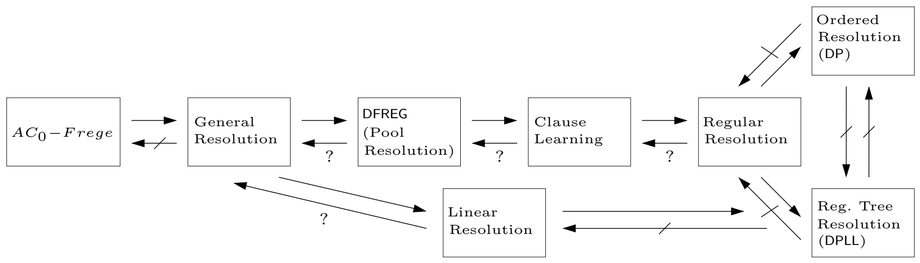

Figure 2: Part of the Proof Complexity Hierarchy which Relates Various Resolution Refinements

Each node in the diagram represents all of the families of formulas which have polynomial size refutations in the system labeling the node. Arrows represent p-simulation relationships between systems. An arrow from system α to system β means that α p-simulates β. A slash through an arrow from α to β represents an exponential separation between β and α. An arrow labeled with a question mark denotes an unknown relationship. Systems to the left in the diagram are generally stronger than systems to the right. Though short refutations exist for larger classes of formulas in stronger systems, finding them is generally more difficult than finding refutations in weaker systems. Hence SAT-solvers based on the Resolution rule generally do not have the full power of RES, but instead implement some form of DPLL which is equivalent to T-RES and is very low in the hierarchy. It is therefore desirable for instances which we want to solve to exist in nodes that are low down in the hierarchy. This gives us a better chance of finding a refutation deterministically in a short amount of time.

We can use this hierarchy to judge the quality of SAT encodings. If some input to an encoding is at one level of the hierarchy and its corresponding output exists only in higher levels, then the encoding is dangerous with respect to that input since its result requires more power to solve. If the input and output of an encoding exist in all of the same levels of the hierarchy, then the encoding is neutral with respect to that input. If some input to an encoding exists nowhere below a certain level in the hierarchy, but its output does, then that encoding actually makes a potentially exponential contribution towards solving the instance. Since every encoding only takes polynomial time to compute, such beneficial encodings can be used as efficient preprocessing steps and identifying them is of great practical interest. We coin the terms *Explosive*, *Stable*, and *Implosive* to refer to encodings which are harmful, neutral, and beneficial with respect to certain families of formulas and certain proof systems. These are defined formally below and examples of each are given.

5.1 Explosivity

Definition 5.1. *Let α be a proof system for a language L_1, let β be a proof system for a language L_2, and let $R : L_1 \rightarrow L_2$ be a reduction from L_1 to L_2. If there exists some family of strings $X = \{x_1, x_2, ...\}$, $X \subseteq L_1$ such that for all k and for all $x_i \in X$ there exists an α-proof P_1 of x_i, but there exists no β-proof P_2 of $R(x_i)$ such that $|P_2| \leq |P_1|^k$, then we say that the reduction R is (α, β)-Explosive on the set X.*

This definition corresponds to our intuitive notion of what constitutes a dangerous reduction, and we can immediately apply it to our main result:

Corollary 5.2. *The Hamiltonian Cycle to SAT reduction above which uses the T, 1, and F clauses is (α,$\mathsf{AC^0}$-Frege)-Explosive on the set containing the K_n^* graphs for any non-Hamiltonicity proof system α which has polynomially-bounded proofs of the K_n^* graphs.*

An example of such a non-Hamiltonicity proof system is NHPS, given in [Her06]. This example is interesting, since tree-like NHPS seems to be weaker than RES, let alone any $\mathsf{AC^0}$-Frege system. We therefore have an example of a reduction that injects enough complexity to send its outputs' difficulty several levels up the proof complexity hierarchy.

Another formal example of Explosivity comes from a corollary of the main result of [HU06a] which proves that the reduction from QBF to Intuitionistic Propositional Logic (IPL) given by Statman in [Sta79] is probably Explosive:

Corollary 5.3. *Unless $\mathcal{NP} = co\mathcal{NP}$, Statman's reduction is (α,$\mathsf{LJ}[\vec{E_S}]$)-Explosive for any QBF proof system α which has polynomially-bounded proofs for any prenex instance of the law of excluded middle (i.e. formulas of the form $p \vee \neg p$).*

Explosivity is caused when an encoding increases the proof complexity of the input instance. In the case of RES, if a reduction fails to introduce clauses which are needed in order to provide a short RES proof, then the reduction is Explosive, and there is no hope of solving the translation. The 'onto' clauses discussed in Section 2 are an example of such clauses which have no short RES derivations themselves and can make an exponential difference to the proof complexity of the reduction's output.

Those interested in proof complexity will note that Explosivity is trivially associated with exponential separations between proof systems. Every example of p-simulation between two proof systems on the same language for which there is a superpolynomial separation implicitly gives an example of (α, β)-Explosivity. If proof system α p-simulates proof system β, but β does not p-simulate α, then the trivial reduction of doing nothing is (α, β)-Explosive on the set of formulas which provides the separation. For this reason, (α, β)-Explosive reductions where α is a strictly stronger proof system than β (for example, ($\mathsf{AC^0}$-Frege, T-RES)-Explosive reductions) are not nearly as interesting as (α, β)-Explosive reductions in which α is a strictly weaker proof system than β.

5.2 Stability

Definition 5.4. *Let α, β, L_1, L_2, and R be as in Definition 5.1. If there exist constants k_1 and k_2 and a family of strings $X = \{x_1, x_2, ...\}$, $X \subseteq L_1$ such that for any α-proof P_1 of x_i there exists a β-proof P_2 of $R(x_i)$ where $|P_2| \leq |P_1|^{k_1}$ and $|P_1| \leq |P_2|^{k_2}$ then we say that the reduction R is (α, β)-Stable on the set X.*

From a proof-complexity point of view, every example of p-equivalence implicitly gives an example of (α, β)-Stability. If α and β are two p-equivalent proof systems for the same language L, then the trivial reduction of doing nothing is both (α, β)-Stable and (β, α)-Stable for the entire language L. For this reason, (α, β)-Stable reductions for p-equivalent proof systems are not nearly as interesting as ones for proof systems for which there is a superpolynomial separation.

Another interesting example of stability is the relationship between RES and Linear Resolution (L-RES) shown by Buresh-Oppenheim and Pitassi in [BOP03]. The authors provide a very simple reduction R which consists of adding trivial clauses of the form $(x_i \vee \neg x_i \vee x_j)$ and $(x_i \vee \neg x_i \vee \neg x_j)$ for all pairs of variables x_i, x_j in the original formula, and show that for every RES refutation of a formula F, there exists an L-RES proof of $R(F)$ which is only polynomially larger. In other words, R is (RES,L-RES)-Stable on all formulas which have polynomially-bounded RES proofs.

Another example of stability comes from Theorem 4.1 above:

Corollary 5.5. *The Hamiltonian Cycle to SAT reduction above which uses the T, O, 1, and F clauses is (α,T-RES)-Stable on the set of K_n^* graphs for any non-Hamiltonicity proof system α which has polynomially-bounded proofs for the K_n^* graphs.*

As already mentioned, NHPS from [Her06] is such an α.

5.3 Implosivity

In practical terms, an even more interesting characteristic for encodings is that of Implosivity. Intuitively, an encoding which takes hard formulas for one proof system and converts them into easy ones for another is Implosive. In other words, Implosive reductions can make otherwise hard instances more accessible to SAT-solvers. Examples of such beneficial reductions are already known to researchers. For example, Kautz and Selman show that SAT encodings of Constraint Satisfiability Problem (CSP) instances can be optimized with respect to local consistency checking and unit propagation [KS03]. In this case the reduction from CSP to SAT actually has beneficial properties, namely that it reduces the proof complexity of its inputs with respect to the consistency conditions. Bailleux and Boufkhad give another good example in [BB03], where they give an encoding that transforms the parity problem, which for many years was considered to be a hard DIMACS instance, into formulas that are easy for DPLL-based solvers.

More formally, the beneficial property of Implosivity is defined as follows:

Definition 5.6. *Let α, β, L_1, L_2, and R be as in Definition 5.1. If there exists some family of strings $X = \{x_1, x_2, ...\}$, $X \subseteq L_1$ such that for all k and for all $x_i \in X$ there exists a β-proof P_2 of $R(x_i)$ but there exists no α-proof P_1 of x_i such that $|P_1| \leq |P_2|^k$, then we say that the reduction R is (α, β)-Implosive on the set X.*

As with Explosivity, Implosivity is trivially associated with p-simulation. Every example of p-simulation between two proof systems on the same language for which there is a superpolynomial separation implicitly gives an example of (α, β)-Implosivity. If proof system β p-simulates proof system α, but α can not p-simulate β, then the trivial reduction of doing nothing is (α, β)-Implosive on the set of formulas which gives the separation. For this reason, (α, β)-Implosive reductions where α is strictly weaker than β (for example, (T-RES, AC^0-Frege)-Implosive reductions) are not nearly as interesting as (α, β)-Implosive reductions in which α is strictly stronger than β.

Again, a non-trivial example of Implosivity comes from the NHPS proof system. Let $G_{\frac{n}{2},\frac{n}{2}}$ be the graph consisting of two disjoint cliques of size $\frac{n}{2}$. These graphs have exponential NHPS lower bounds [Her06]. However, the formulas resulting from applying the reduction from Section 2 which uses the T, O, 1, and F clauses have polynomial T-RES upper bounds [HU06b]. In other words, this reduction is (NHPS, T-RES)-Implosive on the $G_{\frac{n}{2},\frac{n}{2}}$ graphs, which is interesting because the T, 1, F version of the reduction from Corollary 5.2 is (NHPS,AC^0-Frege)-Explosive on the K_n^* graphs. This gives a clear example of how different inputs to the same system can be simplified or complicated depending on encoding.

An interesting potential example of Implosivity is the L-RES reduction R from [BOP03] mentioned above. As already stated, R is (RES,L-RES)-Stable on the entire SAT language. However, it is unknown whether an exponential separation exists between L-RES and RES. If so, then R is (L-RES,L-RES)-Implosive on the inputs which give the separation. Such examples of reflexive implosivity are good candidates for beneficial preprocessing.

Generally speaking, non-trivial Implosivity arises when polytime reductions make use of reasoning which is not available to their target proof system. A polytime reduction in RES might add clauses to the instance which could not otherwise be derived concisely in RES. Given these clauses, Resolution-based solvers can easily solve the problem, but without them, they require exponential time. In effect, such reductions allow solvers to 'cheat' and do work that cannot be done by their underlying proof systems.

5.4 Alternate Hierarchies

Though this the proof system hierarchy may prove to be useful for classifying encodings, we could also produce alternative hierarchies for which the notions of explosivity, stability, and implosivity could be used to classify encodings. In order to use the proof system hierarchy for this task we need to perform a fairly robust analysis of the family of problem instances being studied, as we did in Sections 3 & 4. A more empirical hierarchy based on the real world performance of specific implementations on families of inputs may be preferred.

6 Implications for Proof Complexity

Whenever the relationship between two proof systems on different languages is studied, there must necessarily be a reduction involved. Weak proof systems such as RES and its refinements are not powerful enough to perform polytime reductions. This necessitates the use of a separate polytime algorithm to perform the reduction. Since the details of the reduction can affect the proof complexity of its output, it does not make sense to talk about p-simulation or exponential separation between two weak proof systems over different languages. Rather, one must talk about p-simulation or exponential separation *with respect to a specific reduction.* If a reduction exists which allows one proof system to p-simulate another, we say that the first proof system effectively p-simulates the second. We formally define this notion as follows.

Definition 6.1. *Let $f_1 : S_1^* \to L$ and $f_2 : S_2^* \to L$ be proof systems. If there exists a k and a polytime reduction $r : L_1 \to L_2$ such that $y \in L_1$ if and only if $r(y) \in L_2$ and for all $x_1 \in S_1^*$ there exists an $x_2 \in S_2^*$ such that $r(f_1(x_1)) = f_2(x_2)$ and $|x_2| \leq |x_1|^k$, then we say that f_2 effectively polynomially-simulates f_1.*

If there also exists a polytime computable function $t : S_1^* \to S_2^*$ such that for all $x \in S_1^* r(f_1(x)) = f_2(t(x))$, then f_2 effectively p-simulates f_1.

We can just as easily consider this definition applied to two proof systems over the same language. This yields a generalization of the normal notion of p-simulation. For example, though L-RES is not known to p-simulates RES, it does effectively p-simulate RES since the polytime reduction in [BOP03] is (RES,L-RES)-Stable on the entire SAT language.

7 Concluding Remarks

The idea that encodings can inject complexity into a problem is disconcerting. It is worrisome to think that a reduction from one problem to another can negatively affect the proof complexity of the result and potentially make the instance difficult for proof systems which are located several levels higher in the proof complexity hierarchy than the intended system. Furthermore, as we have shown in this paper, this phenomenon can happen with very natural and even obvious encodings. Even more worrisome is that it does not seem to be at all obvious which types of reductions have this property. With our example, we were lucky enough to see that the input graph was translated to a formula which is very similar to the pigeonhole formulas, but in general we cannot expect to be so lucky. There are probably infinitely many families of formulas which have no short RES proofs and it would not be easy to identify them lurking within the output of an encoding. Random formulas, which are very hard to categorize, as well as other combinatorial problems which have never even been investigated could act very much like the pigeonhole formulas do in our example. If we do not even know what these formulas look like, then it is probably very difficult to predict and avoid reductions which might produce them or something similar to them. Further research is needed in order to characterize which types of reductions have this property.

As a first step towards a characterization, we have outlined a framework for comparing encodings based on the proof complexity hierarchy. The key idea behind the framework is that encodings can affect the proof complexity of the result either beneficially or adversely by overcoming the superpolynomial separation between two proof systems through the use of reasoning that is unavailable to the proof system or by requiring the proof system to derive clauses which cannot be derived concisely.

8 Acknowledgements

We would like to thank Fahiem Bacchus for his many valuable comments and suggestions regarding this paper.

References

[BB03] O. Bailleux and Y. Boufkhad. Efficient CNF Encodings of Boolean Cardinality Constraints. *International Conference on the Principles and Practice of Constraint Programming*, pages 102 – 122, 2003.

[BOP03] J. Buresh-Oppenheim and T. Pitassi. The Complexity of Resolution Refinements. *Proceedings of the 18th Annual IEEE Symposium on Logic in Computer Science*, 2003.

[BT88] S. Buss and G. Turán. Resolution Proofs of Generalized Pigeonhole Principles. *Theoretical Computer Science*, 62:311 – 317, 1988.

[CK01] P. Clote and E. Kranakis. *Boolean Functions and Computation Models.* Springer-Verlag, Berlin, 2001.

[Coo71] S.A. Cook. The Complexity of Theorem-Proving Procedures. *Proceedings of the Third Annual ACM Symposium on the Theory of Computation*, pages 151 – 158, 1971.

[Hak85] A. Haken. The Intractability of Resolution. *Theoretical Computer Science*, 39:297 – 308, 1985.

[Her06] A. Hertel. A Non-Hamiltonicity Proof System. Unpublished Manuscript, 2006.

[HU06a] A. Hertel and A. Urquhart. Proof Complexity of Intuitionistic Propositional Logic. Unpublished Manuscript, 2006.

[HU06b] A. Hertel and A. Urquhart. Prover / Delayer Game Upper Bounds For Tree Resolution. Unpublished Manuscript, 2006.

[KMS96] H. Kautz, D. McAllester, and B. Selman. Encoding Plans in Propositional Logic. *Proceedings of the Fifth International Conference on Knowledge Representation and Reasoning*, 1996.

[KMS97] H. Kautz, D. McAllester, and B. Selman. Ten Challenges in Propositional Reasoning and Search. *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence*, 1997.

[KS03] H. Kautz and B. Selman. Ten Challenges Redux: Recent Progress in Propositional Reasoning and Search. *Ninth International Conference on Principles and Practice of Constraint Programming*, 2003.

[Sta79] R. Statman. Intuitionistic Propositional Logic is Polynomial-Space Complete. *Theoretical Computer Science*, 9:67 – 72, 1979.

[UF96] A. Urquhart and X. Fu. Simplified Lower Bounds for Propositional Proofs. *Notre Dame Journal of Formal Logic*, 37:523 – 545, 1996.