Model-Based Transformations for Quantified Boolean Formulas

Uwe Bubeck

Dissertation

in Computer Science

submitted to the

Faculty of Electrical Engineering,
Computer Science and Mathematics

University of Paderborn

in partial fulfillment of the requirements for the degree of

doctor rerum naturalium
(Dr. rer. nat.)

Paderborn, 2009

Uwe Bubeck
University of Paderborn
Warburger Strasse 100
33098 Paderborn, Germany
uwe@ub-net.de

This is the online version of the corresponding book published by IOS Press and AKA Verlag. The book is available at http://www.iospress.nl/

Please cite as:

Uwe Bubeck.
Model-Based Transformations for Quantified Boolean Formulas.
In: *Dissertations in Artificial Intelligence (DISKI)*, Vol. 329, Wolfgang Bibel (Ed.),
IOS Press, 2010.

ISBN 978-1-60750-545-7

Foreword

Quantified Boolean formulas (*QBF*) are an extension of propositional formulas by allowing universal and existential quantifiers over propositional variables. They are very useful in modeling problems in Artificial Intelligence and Computer Science, for instance to encode planning problems, games and bounded model checking.

Compared to propositional logic, representations by means of quantified Boolean formulas are often significantly shorter, easier to read, and more natural. But the price to be paid for that is an increase of the worst-case complexity of the satisfiability problem. For *QBF*, the satisfiability problem (QSAT) is *PSPACE*-complete, whereas propositional SAT is known to be *NP*-complete.

An extension of *QBF* are so-called dependency quantified Boolean formulas (*DQBF*). These formulas are quantified Boolean formulas with dependency quantification, i.e. for each existential variable we explicitly specify a set of universal variables which the existential variable depends on. This allows not only a clearer notation, but also powerful new encoding techniques.

This thesis addresses - from a theoretical as well as from a practical point of view - methods and techniques for *QBF*, *DQBF* and various relevant subclasses which are helpful for solving the satisfiability problem. It also presents efficient transformations and novel modeling patterns.

The mathematically well-founded results of this work are thoroughly documented, but nevertheless easily comprehensible. They are of particular importance for the development of sophisticated solvers and also in formal methods for modeling problems by means of propositional logic and its extensions.

I strongly recommend Uwe Bubeck's outstanding thesis in particular to all readers interested in research on quantified Boolean formulas.

Hans Kleine Büning

Acknowledgements

A large project like this dissertation could not be possible without the support of many people. First of all, I am especially grateful to my advisor Professor Dr. Hans Kleine Büning. With his invaluable experience, encouragement and patience, he has successfully guided me through this project and has provided me with a stimulating research environment. Furthermore, I would like to thank Professor Dr. John Franco and Professor Dr. Burkhard Monien for reviewing this dissertation and making valuable comments and suggestions. I am also grateful to Professor Dr. Christian Scheideler and Dr. Rainer Feldmann for their interest in my work and for being on my examination committee. In addition, I would like to thank Professor Dr. Ulrich Rückert and Professor Dr. Wilhelm Schäfer for their advice during the initial stages of my research.

Many thanks go to my present and former colleagues in the research group Knowledge-based Systems at the University of Paderborn. Dr. Andreas Goebels, Dr. Oliver Kramer, Dr. Theodor Lettmann and Alexander Weimer have provided valuable advice and have broadened my viewpoint with inspiring discussions on AI and machine learning. I have also enjoyed the good cooperation with Isabela Anciutti, Natalia Akchurina, Markus Eberling, Thomas Kemmerich, Christina Meyer, Dr. Steffen Priesterjahn, Professor Dr. Benno Stein and Yuhan Yan. Special thanks go to Simone Auinger and Patrizia Höfer for their continuous support and for reminding me of a world outside the university. I would like to thank Gerd Brakhane and Friedhelm Wegener for their technical assistance.

I am grateful for the financial and organizational support by the "International Graduate School Dynamic Intelligent Systems" at the University of Paderborn. In particular, I wish to thank Professor Dr. Eckhard Steffen and Astrid Canisius.

Most importantly, I would like to express my deepest gratitude to my parents Christa and Manfred and my sister Sonja. With unconditional love and support, they have accompanied all my studies and made them possible in the first place.

Uwe Bubeck

Contents

1. Introduction

Tremendous progress has been made on algorithms for determining the satisfiability of propositional formulas (SAT). In the last 10-15 years, speed-ups of many orders of magnitude have been achieved by adding intelligent pruning of the search space to the surprisingly simple, but effective DPLL backtracking decision procedure from the early 1960s [DP60, DLL62] that is still at the core of many modern SAT solvers (sometimes complemented by BDD-based symbolic techniques [FKS+04, JS05]). DPLL essentially implements a complete search in a tree-like search space where in the worst case both possible truth assignments are considered for each variable in the formula. The original algorithm already exploits the fact that initial variable assignments quickly imply further assignments on other variables, in particular variables in unit clauses, which makes a further case distinction on those variables unnecessary. Current SAT solvers use sophisticated watched literal data structures [MMZ+01] for efficiently identifying such situations and clever heuristics like VSIDS [MMZ+01] for choosing variables in an order that quickly leads to these inferences. Conflict-driven non-chronological backtracking [MSS96] enables solvers to directly jump back to the causes of conflicts, and subproblems that have proven to be unsatisfiable are avoided in subsequent search by learning suitable conflict clauses [MSS96].

Such techniques appear to be universally useful also for other search or decision problems besides SAT, and modern SAT solvers provide them out of the box in highly optimized implementations. Thus, it is not surprising that it has become quite popular in the last few years to solve difficult decision problems by encoding them as propositional formulas. One of the most prominent examples is bounded model checking (BMC) [BCCZ99, CBRZ01]. Model checking is a technique to automatically verify that a hardware or software system adheres to given requirements. This is accomplished by analyzing the state transition space of the system with respect to a formal specification of the requirements. SAT-based model checking attempts to speed up the process by encoding both the specification checks and the sequential system behavior over a finite interval

of time steps into propositional formulas. Their satisfiability is then determined by a SAT solver. Figure 1.1 illustrates the process from a system-theoretic viewpoint. Another prominent application of SAT-based problem solving is planning [KS92, KS96]. A very recent example is OpenSUSE Linux 11.1 being shipped with a package manager that uses an embedded SAT solver to check package dependencies, and a similar project is in development for Eclipse IDE plugins.

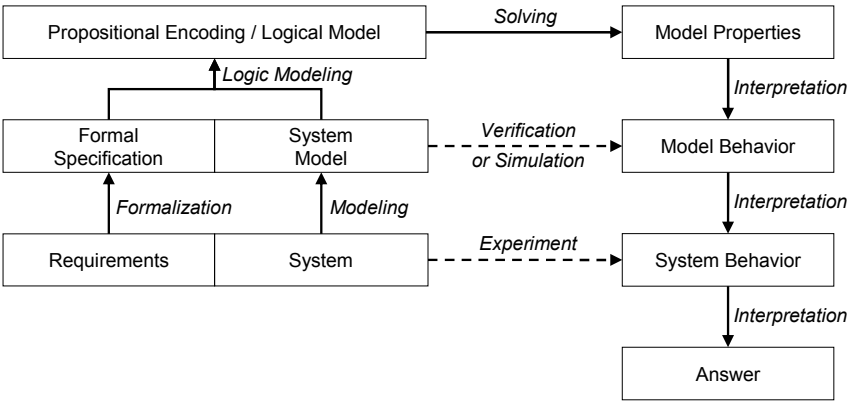

Figure 1.1.: General principle of SAT-based bounded model checking

The success of such applications depends in particular on the quality of the propositional encodings. But finding good encodings can be a difficult task, because propositional logic itself is not expressive enough to model complex situations in a direct way. For example, an important subproblem in bounded model checking is to determine whether a tuple $(\mathbf{v}_0, ..., \mathbf{v}_{2^k})$ of vertices in a directed graph forms a continuous path from $\mathbf{v}_0$ to $\mathbf{v}_{2^k}$. At this point, we only want to have a brief look at this problem, before we cover it in more detail in Section 4.4. We assume that $\mathbf{v}_i$ is represented as a vector of Boolean variables $v_{i,0}, ..., v_{i,n}$, $n \geq k$, which we indicate by writing $\mathbf{v}_i$ in bold. If the transition relation δ of the graph is already given as a propositional formula, the path is continuous if and only if $\delta(\mathbf{v}_i, \mathbf{v}_{i+1})$ is true for all $i = 0, ..., 2^k - 1$. This "for all" cannot be encoded directly in propositional logic and must thus be unrolled as

in the following representation [BCCZ99]:

$$\phi(\mathbf{v}_0,...,\mathbf{v}_{2^k}) := \delta(\mathbf{v}_0,\mathbf{v}_1) \wedge \delta(\mathbf{v}_1,\mathbf{v}_2) \wedge ... \wedge \delta(\mathbf{v}_{2^k-1},...,\mathbf{v}_{2^k})$$

Actual formulas will be substituted for δ, which means it will be hard to recognize the above pattern in the resulting formula. But such an encoding is not only less intuitive. The multiple copies of δ are also very space-consuming, considering that δ represents the whole structure of a possibly very large graph. Then it can easily happen that the resulting formula is by far too voluminous for the capabilities of even the most advanced SAT solvers. In particular, the memory requirements may quickly exceed feasible amounts [ASV+05], which can hardly be offset by allowing more computation time. These observations suggest that we might need a more expressive language for our encodings. Of course, we can expect that this increases the complexity of the corresponding satisfiability problem, but it might allow us to trade time for space.

1.1. Quantification and Expressiveness

Quantified Boolean formulas (*QBF*) generalize propositional formulas by allowing variables to be quantified either universally or existentially, whereas all variables are implicitly existentially quantified in propositional logic. A universally quantified formula $\forall x\, \phi(x)$ is defined to be true if and only if $\phi(0)$ is true *and* $\phi(1)$ is true. Similarly, an existentially quantified formula $\exists y\, \phi(y)$ is true if and only if $\phi(0)$ *or* $\phi(1)$ is true. If free (unquantified) variables are also allowed, we indicate this with a star * and write *QBF**. Quantified Boolean formulas in conjunctive normal form (*CNF*) are denoted *QCNF* or *QCNF** formulas. A detailed definition of the syntax and semantics of quantified Boolean formulas will be given in Chapter 2. In addition, Appendix A provides a brief overview over all classes of formulas which are considered in this work.

Quantification allows natural encodings of problems with inherent forall/exists semantics. The previous example of path connectivity can then be encoded as follows [DHK05]:

$$\Phi(\mathbf{v}_0,...,\mathbf{v}_{2^k}) := \forall\mathbf{u}\forall\mathbf{w} \left(\bigvee_{i=0}^{2^k-1} ((\mathbf{u}=\mathbf{v}_i) \wedge (\mathbf{w}=\mathbf{v}_{i+1})) \right) \rightarrow \delta(\mathbf{u},\mathbf{w})$$

We can observe that the formula now closely matches the informal statement that all successive pairs of vertices must be connected by the transition relation δ, which is more intuitive than the unrolled variant. The formula structure will still be easily recognizable after concrete transition relations have been substituted for δ. Most importantly, this encoding requires only one instance of δ, which makes it much more concise than the propositional representation.

If we only want to know whether two vertices $\mathbf{v}_0$ and $\mathbf{v}_{2^k}$ are connected by some path of length 2^k, we can quantify $\mathbf{v}_1, ..., \mathbf{v}_{2^k-1}$ existentially [DHK05]:

$$\Phi'(\mathbf{v}_0, \mathbf{v}_{2^k}) := \exists \mathbf{v}_1 ... \exists \mathbf{v}_{2^k-1} \forall \mathbf{u} \forall \mathbf{w} \left(\bigvee_{i=0}^{2^k-1} ((\mathbf{u} = \mathbf{v}_i) \wedge (\mathbf{w} = \mathbf{v}_{i+1})) \right) \rightarrow \delta(\mathbf{u}, \mathbf{w})$$

Propositional logic cannot concisely express the fine distinction between these two statements.

Even if the problem to be encoded does not contain any obvious forall/exists semantics, quantification may be used to introduce auxiliary variables that can abbreviate repeating subformulas. For example, consider the following propositional formula:

$$\psi(A, ..., F) := (A \vee \neg B \vee C \vee D) \wedge (A \vee \neg B \vee C \vee \neg E) \wedge (A \vee \neg B \vee C \vee F)$$

Here, the part $A \vee \neg B \vee C$ repeats in multiple clauses. By introducing a new existentially quantified variable y as an abbreviation for $A \vee \neg B \vee C$, we can get a shorter formulation:

$$\Psi(A, ..., F) := \exists y \, (A \vee \neg B \vee C \vee \neg y) \wedge (y \vee D) \wedge (y \vee \neg E) \wedge (y \vee F)$$

The important point to notice is that both formulas ψ and Ψ have exactly the same free variables. Furthermore, both formulas exhibit the same truth value for the same truth assignments to the free variables, so we can treat the two formulas as equivalent. In propositional logic, it is also possible to add new (implicitly existential) variables in a similar way (the technique is well known from the Tseitin transformation [Tse70]), but the resulting formula is usually only satisfiability-equivalent to the original formula in the propositional case.

Universally quantified auxiliary variables allow the abbreviation of multiple instantiations of a single subformula with different arguments, a technique that is not directly possible in propositional logic. An example is the above path

connectivity encoding Φ which compresses multiple instantiations of the subformula δ for different arguments $(\mathbf{v}_0, \mathbf{v}_1), ..., (\mathbf{v}_{2^k-1}, \mathbf{v}_{2^k})$. With clever combinations of universally and existentially quantified auxiliary variables, quantified Boolean formulas can effectively condense structures that show repetitions or similarity. A small compilation of the most important ideas is presented as part of the fundamentals description in Section 2.6.

In [JB07], different bounded model checking encodings are compared, with the *QBF** representations often being drastically more compact than the best propositional variants. For example, the best propositional encoding of the instance "periodic.N" from the CMU SMV distribution has a size of 28.1 megabytes for induction depth $k = 96$, but only 2.1 MB in *QBF**. Similarly, [ASV$^+$05] considers hierarchical debugging with propositional and *QBF* encodings, where the largest example requires more than 300 MB of memory in the propositional case, versus 57 MB for *QBF*, with the gap growing asymptotically as the examples are getting larger.

As suggested earlier, there is unfortunately a price to pay for the compactness of *QBF**. Determining the satisfiability of formulas in *QBF* or *QBF** is *PSPACE*-complete, which is assumed to be significantly harder than the *NP*-completeness of the propositional SAT problem. On the other hand, many interesting verification problems are also *PSPACE*-complete, e.g. propositional linear temporal logic (*LTL*) satisfiability [SC85] or symbolic reachability in sequential circuits [Sav70]. Thus it seems quite natural to solve such problems by encoding them as quantified Boolean formulas. Furthermore, *QBF** solving in practice has made impressive progress over the last few years. One reason is that some of the most successful improvements of propositional SAT have been lifted to *QBF** as well. Examples include Q-resolution [FKKB95], watched literal data structures [GGN$^+$04] or conflict and solution learning [GNT02]. In addition, clever new approaches specific to *QBF** have been suggested, such as symbolic skolemization [Ben05a], universal expansion [Bie05] or parallel search techniques [FMS00].

Some of these operations, e.g. Q-resolution, are difficult to handle without additional restrictions on the occurrence of quantifiers. It is therefore generally assumed that all quantifiers occur at the beginning of the formula in a distinct *prefix*, followed by the propositional *matrix*. Interestingly, the hardness of quantified Boolean formulas appears to depend in particular on the complexity of the quantifier prefix, and that holds from both a practical perspective as well as

from a complexity-theoretic viewpoint. For a hierarchy of complexity classes, the so-called *polynomial-time hierarchy*, it has been shown by Stockmeyer and Wrathall [Sto76, Wra76] that there is a close relationship between the levels in this hierarchy and the number of quantifier alternations in the prefix of quantified Boolean formulas (for more details, see Section 2.3). It is widely believed that complexity classes from a lower level of the hierarchy are properly included in classes from higher levels, which means the decision problem of quantified Boolean formulas appears to become more difficult with each additional block of quantifiers.

From a practical perspective, a purely existentially quantified Boolean formula can be solved with one call to an ordinary SAT solver. The addition of a universal quantifier requires considering both possible truth values for it, so that it might be necessary to make two calls to the SAT solver. Similarly, an extension of DPLL to *QBF** may need to recursively solve both subproblems by branching on the universal. At the same time, the variable ordering imposed by the nesting of the quantifiers must be respected. In a formula with prefix $\exists x \forall y \exists z$, z may be assigned different values depending on the value of y, but the value of x must be chosen first and is the same for all values of y. That means we cannot branch on y before assigning x, which severely limits the effectiveness of variable selection heuristics in (Q)DPLL-based solvers. We will later see that approaches like universal expansion or symbolic skolemization are also affected by more difficult prefixes.

1.2. Thesis Goals and Contributions

Despite the obvious progress that *QBF** solvers have made, there is still a clear performance gap between them and their SAT brethren, which is not really surprising given the additional complexity described in the previous paragraph. That means *QBF** can be seen as a tradeoff between computation time on the one hand and benefits like more natural and more compact encodings on the other hand. More natural encodings are probably less error-prone and require less engineering time. Most importantly, more compact encodings allow solving larger problems for which propositional approaches exceed feasible memory bounds. However, this is not a $0/1$ decision for or against quantification. Instead, it appears that more compact encodings require more quantifiers and more alternations of quantifier blocks. For the example of continuous paths in graphs, we

will later encounter a second QBF^* encoding that is even more concise than the one presented above, but on the other hand requires an unbounded number of quantifier alternations.

In general, however, it is not well understood how the complexity of the prefix is related to the expressiveness of the corresponding class of formulas, and how that relationship is influenced by the formula structure. Besides assumptions about the polynomial-time hierarchy, only concrete examples and some general encoding techniques like the ones mentioned in the previous section are available. It is important to point out that by expressive power or expressiveness, we mean the ability to provide *short* encodings. This differs from descriptive complexity theory [Imm99], where expressiveness usually considers the general ability to represent a given property by a certain class of formulas. In that sense, quantified Boolean formulas are equally powerful as propositional logic, because quantifiers are only abbreviations which can always be expanded as explained below. But that may cause a blowup in formula size, which is why we focus specifically on the size of encodings. We are convinced that a better understanding of the expressive power of quantification in this respect is necessary for efficient and successful applications of QBF^*. There are in particular three questions in which we are interested:

1. How does the expressiveness of quantifiers depend on the structure of the propositional matrix of the formula?
2. Is it possible to simplify given quantifier prefixes without a significant loss of compactness?
3. Which usage patterns of quantifiers lead to encodings with a good tradeoff between compactness and complexity?

Questions 2 and 3 appear to be closely related: results that allow the simplification of given formulas should also be helpful as guidelines for suitable new encodings. Question 1 leads to the most important contribution of this thesis. We identify relationships between the structure of the formula matrix and the expressiveness of quantifiers, and we show how these relationships can be used for suitable transformations into simplified representations, which establishes a connection between the first question and the other two.

We describe the behavior of quantifiers by Boolean function models (originally introduced in [KBSZ04] and explained in detail in Section 2.7). Similar to Skolem functions in predicate logic, an existentially quantified variable y_i is

mapped to a Boolean function f_{y_i} over those universal variables whose quantifiers precede the quantifier of y_i and over the free variables (if applicable). These functions f_{y_i} are represented as propositional formulas. The goal is to characterize the expressive power of an existential quantifier by specifying the structure of its associated model function. Conversely, the role of a universal quantifier can be determined by investigating its occurrences in model functions of existentially quantified variables. Existing work shows that the simple classes of quantified *2-CNF* formulas and quantified Horn formulas without free variables have models of a very restricted structure [KBSZ04, KBZ05]. We contribute results about more complicated classes like quantified Horn formulas with free variables or *QCNF** formulas.

Boolean function models can obviously be used to eliminate existential quantifiers $\exists y_i$ by replacing all occurrences of y_i in the matrix with the corresponding propositional representation of f_{y_i}. However, this requires computing the exact (and potentially exponentially large) specification of the model function, not just some general properties about its structure. Can another technique of quantifier elimination make better use of information about the structure of model functions? Quantification in *QBF** is defined according to the equivalences $\exists y\ \Phi(y,\mathbf{z}) \approx \Phi(0,\mathbf{z}) \vee \Phi(1,\mathbf{z})$ (the quantified version of the well-known Shannon Expansion for propositional logic) and the dual *universal expansion* $\forall x\ \Phi(x,\mathbf{z}) \approx \Phi(0,\mathbf{z}) \wedge \Phi(1,\mathbf{z})$. By using these equivalences, a quantifier can be expanded by duplicating the remaining formula. Care must be taken to duplicate also subsequent quantifiers that are in the scope of the expanded quantifier.

It is clear that both expansions may cause exponential growth when applied repeatedly. But for *QCNF** formulas, the expansion of an existential quantifier usually requires a retransformation into *CNF*, which might make the resulting formula much larger. We will later see that Q-resolution as an alternative to eliminate existential quantifiers is also very expensive in general. We make the important observation that for clausal formulas, universal quantifiers are typically cheaper to eliminate, although their expansion is still exponential in the worst case. Conversely, this suggests that universal quantification itself is slightly less powerful in *CNF* formulas than existential quantification. This is also backed by the obvious simplicity of the satisfiability problem for purely universally quantified *QCNF* formulas. We will later show that the imbalance between universal and existential quantification is particularly obvious for Horn formulas and generalizations thereof. Accordingly, we focus specifically on the expansion of universals in clausal formulas. One of our central ideas is to simplify universal

expansion by exploiting restrictions on the structure of model functions, which in turn can be established by considering the formula structure.

The remainder of this section provides a brief overview of our main results in a more informal style. Fully detailed results, along with lots of others not mentioned here, and discussions of existing work are given in the corresponding main chapters.

One particular focus of our work is on quantified Horn formulas, that is, formulas in which each clause contains at most one positive literal. How does this restriction influence the behavior of the quantifiers? Detailed characterizations of function models for formulas without and with free variables allow us to prove that the behavior of the existential quantifiers depends only on the cases in which at most one of the universally quantified variables is zero. By considering only these cases, universal expansion can be simplified dramatically. The result even holds if the free variables do not satisfy the Horn property.

Main Result 1. *$QCNF^*$ formulas with unrestricted free literals and at most one positive quantified literal per clause ($QHORN^b$) can be transformed in quadratic time into equivalent purely existentially quantified formulas of quadratic length.*

This class of formulas has interesting applications. We demonstrate that it can be used to encode graph structures and propositional *CNF* transformations in a natural way. The latter implies that the class $QHORN^b$ is expressive enough to represent arbitrary (i.e. non-*CNF*) propositional formulas in polynomial length. We also obtain a new compact graph-based *CNF* transformation that nicely preserves and visualizes the structure of the propositional input formula.

We believe that quantification is most valuable in those cases where quantifiers have a global impact on the whole formula. Local quantifiers might still be useful for encoding problems in a natural way, but they do not contribute much to the compactness of the encoding, yet they can make solving the formulas much harder in practice. Accordingly, we suggest a preprocessing for $QCNF^*$ formulas which attempts to eliminate as many of those cheap quantifiers as possible within given bounds. Again, we focus on universal expansion and apply Q-resolution specifically to reduce the costs of universal expansion.

Main Result 2. *We present a preprocessing of QCNF* formulas by expanding a suitable selection of universally quantified variables with bounded expansion costs. Experiments with well-known problems from the QBFLIB formula collection demonstrate that this preprocessing is very effective on simplifying the prefixes and can significantly improve the performance of state-of-the-art QBF solvers.*

A problem with quantified Boolean formulas is the enforcement of a linear arrangement of quantifiers in the prefix. That makes it usually unavoidable to place existential quantifiers within the scope of semantically unrelated universal quantifiers. Of course, it is not necessary for a solver to consider different values for such existentials depending on assignments to unrelated universals. But how can a solver know which existentials really depend on a given universal?

Main Result 3. *In QCNF* formulas, compact sets of dependent existentials can be obtained by computing transitive closures of local connectivity between variables in common clauses in consideration of variable polarity. In the best case, this can reduce the number of dependent existentials, and thus the arity of model functions, by an arbitrarily large factor in comparison to existing approaches.*

An alternative to recovering variable dependencies from given formulas is to extend quantified Boolean formulas with explicitly given dependencies, known as *dependency quantified Boolean formulas* (*DQBF* or *DQBF** with free variables). This allows new modeling patterns, which we demonstrate with an encoding of path connectivity that is even more compact than the previously mentioned *QBF** encodings. While dependency quantification generally causes another increase in complexity, we consider easier subclasses defined by restrictions on the prefix structure. In addition, we show that important techniques like universal quantifier expansion and our results on Horn formulas can be lifted naturally to these formulas.

Main Result 4. *Dependency quantified Horn formulas DQHORN* constitute a tractable subclass of DQBF*.*

1.3. Document Structure

Before we delve into our investigations, we briefly review some of the theory of quantified Boolean formulas in Chapter 2. In particular, we provide a collection of encoding techniques and introduce Boolean function models.

Our contributions are distributed onto three main chapters. Chapter 3 focuses on quantified Horn formulas without and with free variables and the previously mentioned generalization $QHORN^b$. Chapter 4 considers dependency quantified Boolean formulas that can explicitly indicate variable dependencies. Chapter 5 begins with a theoretical part that refines universal expansion in $QCNF^*$ formulas and then presents our preprocessing approach based on bounded universal expansion.

Each of these main chapters starts with a short abstract, which can safely be skipped by the reader if a more gentle introduction to the topic is desired. The abstracts also contain pointers to own peer-reviewed publications on which the corresponding chapter is based. The actual text then begins with a motivation section. The second section is always "Research Goals and Related Work", where we introduce our contributions on the topic and show how they are connected to existing work. Each chapter also has its own conclusion, which is complemented in Chapter 6 by a brief global summary with suggestions for future work.

For quick reference, Appendix A provides a collection of the definitions of all formula classes which are considered in this document. Appendix B contains an abstract of the whole thesis.

2. Fundamentals

In this chapter, we recall the necessary basic concepts and terminology of quantified Boolean formulas and clarify the notation that we are going to use. In addition, we provide a collection of encoding techniques and introduce Boolean function models. For a more detailed introduction to quantified Boolean formulas, we refer the reader to [KBB09, KBL99].

2.1. Syntax and Semantics

A quantified Boolean formula is a propositional formula or a formula of the form $\forall x\, \phi(x)$ or $\exists y\, \phi(y)$ where x, y are propositional variables and ϕ is a propositional or quantified Boolean formula. $\forall x\, \phi(x)$ is defined to be true if and only if $\phi(0)$ is true *and* $\phi(1)$ is true. Variables which are bound by universal quantifiers are called *universal variables* and are usually given the names $x_1, ..., x_n$. Similarly, $\exists y\, \phi(y)$ is defined to be true if and only if $\phi(0)$ *or* $\phi(1)$ is true. In this case, y is called an *existential variable*. Those usually have names $y_1, ..., y_m$.

In our notation, we assume that the logical connectives have a higher binding priority than the quantifiers, so we can leave out parentheses if a quantifier is supposed to span both operands of a binary connective: $\forall x\, \phi \wedge \psi := \forall x\, (\phi \wedge \psi)$. On the other hand, we need parentheses to indicate that the scope of a quantifier covers only one subformula, e.g. $(\forall x\, \phi) \wedge \psi$.

A quantified Boolean formula Φ is in *prenex form* if $\Phi = Q_1 v_1 ... Q_k v_k\, \phi$ with quantifiers $Q_i \in \{\forall, \exists\}$ and a propositional formula ϕ. We call $Q := Q_1 v_1 ... Q_k v_k$ the *prefix* and ϕ the *matrix* of Φ. Unless mentioned otherwise, we assume that *QBF* formulas are always in prenex form.

The matrix of a quantified Boolean formula often has a particular structure. It is in *negation normal form* (*NNF*) if and only if every negation occurs immediately

in front of a variable. A *literal* is then a positive propositional variable (v) or a negated variable ($\neg v$), and a *clause* $C = l_1 \vee ... \vee l_h$ is a disjunction of literals. Since the order of literals in a clause is irrelevant, we can also consider a clause as a set $C = \{l_1, ..., l_h\}$ of literals. A formula is in *conjunctive normal form* (*CNF*) if and only if it is a conjunction of clauses $\phi = C_1 \wedge ... \wedge C_q$. Again, we also use set notation $\phi = \{C_1, ..., C_q\}$. The dual *disjunctive normal form* (*DNF*) denotes a disjunction of conjunctions of literals. Prenex formulas with arbitrary matrix can be transformed into one of these normal forms by applying propositional equivalences like the laws of associativity, distributivity or De Morgan on the matrix. For a given formula class or normal form K, we denote by QK the class of quantified Boolean formulas whose matrix belongs to K. For example, *QCNF* is the class of *QBF* formulas with matrix in *CNF*, and *QDNF* denotes formulas with matrix in *DNF*. We use italics for formula and complexity classes.

Variables which are not bound by quantifiers are *free* variables. Formulas without free variables are said to be *closed*. If free variables are allowed, we indicate this with an additional star * after the name of the formula class. Accordingly, *QBF* is the class of closed quantified Boolean formulas, and QBF^* denotes quantified Boolean formulas with free variables (analogously, we have *QCNF* and $QCNF^*$, etc.). We write $\Phi(z_1, ..., z_r) = Q\ \phi(z_1, ..., z_r)$ or $\Phi(\mathbf{z}) = Q\ \phi(\mathbf{z})$ for a QBF^* formula with prefix Q, matrix ϕ and free variables $\mathbf{z} = (z_1, ..., z_r)$. We denote by $vars(\Phi)$ the set of all variables that occur in Φ, and $freevars(\Phi)$ contains all free variables in Φ.

A closed *QBF* formula is either true or false. It is true if and only if there exists an assignment of truth values to the existential variables depending on the preceding universal variables such that the propositional matrix of the formula is true for all values of the universal variables. For example, $\Phi = \forall x \exists y\ (x \vee y) \wedge (\neg x \vee \neg y)$ is true, because by choosing $y = 1$ when $x = 0$ and $y = 0$ when $x = 1$, we can satisfy the formula for all values of x. More formally, we can substitute the formula $f(x) = \neg x$ for all occurrences of y in the matrix of Φ, written as $\Phi[y/f]$, such that the resulting matrix $(x \vee \neg x) \wedge (\neg x \vee x)$ is tautological.

The truth value of a QBF^* formula depends on the value of the free variables. A QBF^* formula $\Phi(\mathbf{z})$ is *satisfiable* if and only if there exists a truth assignment $t(\mathbf{z}) := (t(z_1), ..., t(z_r)) \in \{0,1\}^r$ to the free variables $\mathbf{z} = (z_1, ..., z_r)$ for which $\Phi(t(\mathbf{z}))$ is true. Here, $\Phi(t(\mathbf{z}))$ denotes the closed *QBF* formula that results from substituting the truth values $t(z_1), ..., t(z_r)$ for the free variables $z_1, ..., z_r$. For example, the formula $\Phi(z) = \forall x \exists y\ (x \vee y \vee \neg z) \wedge (\neg x \vee \neg y) \wedge (\neg y \vee z)$ is satisfiable:

for $z = 1$, we can choose $y = \neg x$, and substituting z and y produces the tautological matrix $(x \vee \neg x \vee 0) \wedge (\neg x \vee x) \wedge (x \vee 1)$. In that particular example, the formula is also true for $z = 0$: in this case, we can choose $y = 0$, and all clauses are satisfied. We could combine both cases and say that we choose $y = \neg x \wedge z$. Such an assignment will later be called an *equivalence model function*.

2.2. Basic Concepts and Notation

Two QBF^* formulas $\Psi_1(z_1,...,z_r)$ and $\Psi_2(z_1,...,z_r)$ are said to be *equivalent* ($\Psi_1 \approx \Psi_2$) if and only if $\Psi_1 \models \Psi_2$ and $\Psi_2 \models \Psi_1$, where *semantic entailment* $\models$ is defined as follows: $\Psi_1 \models \Psi_2$ if and only if for all truth assignments $t(\mathbf{z}) = (t(z_1),...,t(z_r)) \in \{0,1\}^r$ to the free variables $\mathbf{z} = (z_1,...,z_r)$, it holds that $\Psi_1(t(\mathbf{z})) = 1 \Rightarrow \Psi_2(t(\mathbf{z})) = 1$.

A weaker form of equivalence is *satisfiability equivalence*, where the two formulas may have different free variables. Two QBF^* formulas $\Psi_1(z_1,...,z_r)$ and $\Psi_2(z'_1,...,z'_s)$ are *satisfiability-equivalent* ($\Psi_1 \approx_{SAT} \Psi_2$) if and only if it holds that Ψ_1 is satisfiable $\Leftrightarrow$ Ψ_2 is satisfiable. Notice that equivalence and satisfiability equivalence coincide for closed formulas: $\Psi_1 \approx \Psi_2 \Leftrightarrow \Psi_1 \approx_{SAT} \Psi_2$ for $\Psi_1, \Psi_2 \in QBF$.

Without loss of generality, we assume that all variable names in a prenex QBF^* formula $\Phi(\mathbf{z}) = Q\ \phi(\mathbf{z})$ are unique, that is, no variable appears twice in the prefix Q. We call successive quantifiers of the same kind in Q a *quantifier block*. Blocks are defined to be maximal, such that subsequent blocks S_i and S_{i+1} are always associated with different kinds of quantifiers. That means if Q has the form $Q = \forall x_{1,1}...\forall x_{1,n_1}\exists y_{1,1}...\exists y_{1,m_1}...\forall x_{r,1}...\forall x_{r,n_r}\exists y_{r,1}...\exists y_{r,m_r}$ with $n_i \geq 1$ and $m_i \geq 1$ for $i = 1,..,r$, we simply write $Q = \forall X_1 \exists Y_1 ... \forall X_r \exists Y_r$ with quantifier blocks $X_i = (x_{i,1},...,x_{i,n_i})$ and $Y_i = (y_{i,1},...,y_{i,m_i})$, $i = 1,..,r$. According to their sequence in the prefix, quantifier blocks are ordered linearly $S_1 < ... < S_s$. We call S_s the *innermost* and S_1 the *outermost* block. Occasionally, we also treat the blocks as sets and apply the basic set operations and relations on them. That allows us to write expressions like $v \in X_r \cup Y_r$, which means v is a variable that is bound in the innermost existential or universal quantifier block.

The order of the quantifier blocks induces a partial order on the variables. Let l_1 and l_2 be two literals in Φ, then we define $l_1 < l_2$ if the variable in l_1 occurs in a

quantifier block which precedes the block in which the variable of l_2 appears. If both variables occur in the same block, the order of the literals is undefined. We say that a universally quantified variable x_i *dominates* an existential variable y_j if and only if $x_i < y_j$. Equivalently, we can also say that y_j *depends on* x_i.

With $|\Phi|$, we denote the *length* of a quantified Boolean formula Φ, which is defined to be the number of occurrences of variable symbols, including the quantified variables in the prefix. For example, $\Phi(z) = \forall x \exists y\ (x \vee y) \wedge (\neg x \vee \neg y) \wedge (\neg y \vee z)$ has length $|\Phi| = 8$.

Some additional useful notation is the following:

For a propositional formula f and $\varepsilon \in \{0,1\}$, we let $f^{\varepsilon} := \begin{cases} \neg f & ,\text{if } \varepsilon = 0 \\ f & ,\text{if } \varepsilon = 1 \end{cases}$.

With $AB := (a_1,...,a_m,b_1,...,b_n)$, we denote the concatenation of two tuples $A = (a_1,...,a_m)$ and $B = (b_1,...,b_n)$.

2.3. Subclasses and Complexity Results

We have already pointed out in the introduction that QBF^* satisfiability is a *PSPACE*-complete problem [MS73]. It is easy to see that $QCNF^*$ and even $QDNF^*$ are still *PSPACE* complete. But there are also interesting subclasses of QBF^* that belong to a lower or even tractable complexity class. Such subclasses are typically defined by restrictions on the structure of the prefix or by restrictions on the formula structure.

An obvious restriction on the structure of the prefix is to limit the number of different prefix blocks. For $k \in \mathbb{N}$, we let the *prefix type* Σ_k denote QBF^* formulas with a prefix that has k quantifier blocks and begins with an existential one. Analogously, Π_k is the class of QBF^* formulas with k prefix blocks where the outermost block is universal. $\Sigma_0 = \Pi_0$ denote the class of propositional formulas. For example, $\Phi = \exists y_1 \forall x \exists y_2\ \phi$ with a propositional matrix ϕ is a formula in Σ_3.

These prefix types are closely related to classes of the polynomial-time hierarchy which was introduced in [MS72] as follows for $k \geq 0$:

$$\Delta_0^P := \Sigma_0^P := \Pi_0^P := P$$
$$\Sigma_{k+1}^P := NP^{\Sigma_k^P},\ \Pi_{k+1}^P := \text{co-}\Sigma_{k+1}^P,\ \Delta_{k+1}^P := P^{\Sigma_k^P}$$

Here, NP^A (respectively P^A) is the class of problems which can be decided in polynomial time by a nondeterministic (respectively deterministic) Turing machine augmented by an oracle for a complete problem in A. Obviously, classes on higher levels include classes from lower levels, e.g. $\Sigma_k^P \subseteq \Delta_{k+1}^P \subseteq \Sigma_{k+1}^P$. It is commonly assumed that these inclusions are proper.

In [Sto76, Wra76], it has been shown that the satisfiability problem for quantified Boolean formulas with prefix type Σ_k is Σ_k^P-complete ($k \geq 1$). Analogously, it is Π_k^P-complete for Π_k formulas ($k \geq 1$).

Restrictions on the formula structure are well-known from propositional logic. In this work, Horn formulas play a prominent role. The class of *propositional Horn formulas* (*HORN*) contains all *CNF* formulas with at most one positive literal per clause. For a constant $k \geq 2$, *k-HORN* means *HORN* formulas with at most k literals per clause.

With *QHORN* (*Qk-HORN*, respectively), we denote *quantified (k-)Horn formulas*, that means formulas

$$\Phi = Q_1 v_1 ... Q_k v_k \ \phi(v_1, ..., v_k)$$

where $\phi \in$ *HORN* (*k-HORN*, respectively). Formulas

$$\Psi(z_1, ..., z_r) = Q_1 v_1 ... Q_k v_k \ \psi(v_1, ..., v_k, z_1, ..., z_r)$$

with free variables which also satisfy the Horn property, that is $\psi \in$ *HORN* (*k-HORN*, respectively), are called *quantified (k-)Horn formulas with free variables*, abbreviated *QHORN** (*Qk-HORN**, respectively).

From the quantified version of Schaefer's dichotomy theorem [Sch78] (fully proven only later, e.g. in [Dal97, CKS01]), it follows that exactly the following local restrictions on the structure of clauses in *QCNF** formulas produce tractable subclasses:

- *Horn* clauses (at most one positive literal)
- *anti-Horn* clauses (at most one negative literal)
- *Krom* clauses (at most two literals)
- *XOR* clauses (clauses of the form $x_1 \oplus ... \oplus x_n = 1$ or $x_1 \oplus ... \oplus x_n = 0$)

Of these subclasses, Horn formulas appear to be the most interesting and the most useful. We will study them comprehensively in Chapter 3.

2.4. Simplification Techniques

There are various standard equivalence-preserving simplification techniques for *QCNF** formulas. *Forall reduction* or *Universal reduction* [KBL99] allows us to remove from a non-tautological clause $C = \varphi \vee x^\varepsilon$ a universal literal x^ε if x does not dominate any existential variable in φ. If φ contains no existential or free variables (C is purely universal), we get an empty clause, and Φ is unsatisfiable. Consider the example $\exists y_1 \forall x \exists y_2\ (\neg y_1 \vee \neg x) \wedge (y_1 \vee \neg x \vee y_2) \wedge (z \vee x)$. In the first clause, x does not dominate y_1 and can be removed. In the second clause, x dominates y_2, so the clause remains unchanged. In the last clause, z is a free variable, but there is no existential variable dominated by x, so x can be deleted. We have the resulting formula $\exists y_1 \forall x \exists y_2\ \neg y_1 \wedge (y_1 \vee \neg x \vee y_2) \wedge z$.

Unit propagation and *pure literal detection* are well known for propositional formulas. They can also be applied to *QCNF** formulas [CSGG02]:

1. If Φ contains a clause $C = y^\varepsilon$ with a single existential literal y^ε, then logical equivalence is preserved when we assign $y = \varepsilon$ by replacing all occurrences of y^ε in ϕ with 1 and all occurrences of $y^{1-\varepsilon}$ with 0. In the forall-reduced formula from the last paragraph, $\neg y_1$ is an existential unit which can be propagated. We obtain the formula $\forall x \exists y_2\ (\neg x \vee y_2) \wedge z$.
2. If Φ contains a quantified variable v which occurs only positively or only negatively in ϕ, then logical equivalence is preserved in the following cases:

   a) if v is existentially quantified and all clauses containing v are deleted from Φ.

   b) or if v is universally quantified and the occurrences of v are removed from all clauses of ϕ.

   In the formula from Item 1, y_2 occurs only positively, so we can delete the first clause. We can conclude that the formula is equivalent to just z.

We say that a clause C_i *subsumes* another clause C_j if and only if every literal in C_i also occurs in C_j, that means $C_i \subseteq C_j$ when considering clauses as sets. Then it is clear that the satisfiability of C_i of immediately implies the satisfiability of C_j, thus we can remove such subsumed clauses from a *QCNF** formula.

Another standard technique is the elimination of *dual binary clauses*. A clause $C_i = l_1 \vee ... \vee l_k$ is *dual* to another clause C_j if and only if C_j consists of the

negated literals in C_i, that is $C_j = \neg l_1 \vee ... \vee \neg l_k$. If a *QCNF** formula Φ contains a pair of dual binary clauses over variables v_1 and v_2, we can conclude that $v_1 \leftrightarrow v_2$ or $v_1 \leftrightarrow \neg v_2$. In the first case, every occurrence of v_2^ε in Φ can be replaced with v_1^ε, in the second case v_2^ε with $v_1^{1-\varepsilon}$, always assuming that $v_1 < v_2$ or both occur in the same quantifier block. If $v_2 < v_1$, both substitutions are performed in the inverse direction. This guarantees that no additional degrees of freedom are introduced. To preserve equivalence, we also need the prerequisite that all clauses in Φ are forall-reduced, so that the procedure only instantiates existentials, but not universals. For example, consider the following *QCNF** formula: $\Phi = \forall x \exists y\ (x \vee \neg y) \wedge (\neg z \vee x \vee y) \wedge (\neg x \vee y)$. The first clause is dual to the last one, so $x \leftrightarrow y$ and $\Phi \approx \forall x\ (x \vee \neg x) \wedge (\neg z \vee x \vee x) \wedge (\neg x \vee x) \approx \neg z$.

For more details on the above simplifications and some hints on efficient implementations, we refer the reader to [CSGG02, Bie05].

2.5. Q-Resolution

Q-resolution [FKKB95] extends the concept of propositional resolution to quantified Boolean formulas. Resolution is based on the idea of combining a clause with a positive literal l with another clause that contains $\neg l$, so that the complementary literals l and $\neg l$ disappear. In Q-resolution, we resolve on literals over free or existentially quantified variables.

Definition 2.5.1. *(Q-Resolution)*
*Let $\Phi = Q\phi$ be a formula in QCNF* with matrix ϕ and prefix Q. Let ϕ_1 and ϕ_2 be non-tautological clauses of ϕ, where ϕ_1 contains the existential or free literal y and ϕ_2 contains the literal $\neg y$. We obtain a **Q-resolvent** σ of ϕ_1 and ϕ_2 by applying the following steps (1) to (3):*

1. *For $i = 1,2$, eliminate all occurrences of universal literals in ϕ_i which do not dominate any existential literal which occurs in ϕ_i. The resulting clauses are denoted by ϕ_1' and ϕ_2'.*
2. *Eliminate all occurrences of y in ϕ_1' and all occurrences of $\neg y$ in ϕ_2'. We obtain ϕ_1'' and ϕ_2''.*
3. *$\sigma := \phi_1'' \vee \phi_2''$.*

We write $\Phi \mid\!\frac{1}{\text{Q-Res}}\, Q(\phi \wedge \sigma)$ *or* $\Phi \mid\!\frac{1}{\text{Q-Res}}\, \sigma$*, and* $\mid\!\frac{*}{\text{Q-Res}}$ *denotes the reflexive and transitive closure of Q-resolution derivability.*

The elimination of universals which do not dominate any existential in the same clause (Item 1 in Definition 2.5.1) has already been presented independently of Q-resolution as the universal reduction simplification rule. We must include it here, because Q-resolution is only refutation complete for arbitrary $QCNF^*$ formulas if universal reduction is applied in each resolution step. In the unsatisfiable formula

$$\Phi = \exists y_1 \forall x_1 \forall x_2 \exists y_2\ (y_1 \vee x_1 \vee y_2) \wedge (\neg y_1 \vee \neg x_1 \vee y_2) \wedge (x_2 \vee \neg y_2)$$

initial universal reduction cannot yield any simplifications, and we cannot resolve on y_1 due to the tautological occurrences of x_1 and $\neg x_1$. We call such complementary literals *blocking universals*. If we resolve on y_2 instead, we obtain the two resolvents $(y_1 \vee x_1 \vee x_2)$ and $(\neg y_1 \vee \neg x_1 \vee x_2)$. Now we can only resolve on y_1 if we first apply universal reduction on both resolvents, which produces the simplified clauses (y_1) and $(\neg y_1)$ that subsequently resolve to the empty clause.

The previous example also shows that the exchange lemma for resolution in propositional formulas [KBL99] no longer holds for $QCNF^*$. In the example, resolving first on y_2 and then on y_1 leads to the empty clause, whereas direct Q-resolution on y_1 leads to a tautological clause. Such blockages usually require a detour of several steps.

It is well known that unit resolution is refutation complete for propositional Horn formulas. This restriction can be adapted in a straightforward way to the quantified case. *Q-unit resolution* is Q-resolution where one of the parent clauses is required to be an $\exists$-*unit clause*. A clause is said to be $\exists$-unit if and only if it contains at most one free or existentially quantified literal in addition to an arbitrary number of universal literals. [FKKB90] has shown that Q-unit resolution is refutation complete for a generalization of quantified Horn formulas called $QEHORN^*$, which includes all $QCNF^*$ formulas that are conjunctions of Horn clauses when ignoring the universally quantified literals. Without loss of refutation completeness, the $\exists$-unit requirement can be augmented by requiring the $\exists$-unit clause to contain a *positive* free or existential literal, which is the so-called *Q-pos-unit resolution* [KBL99].

It is possible to eliminate an existential quantifier by performing all possible Q-resolutions on it [DP60, KBL99, Bie05].
Consider the example $\Phi(z) = \forall x_1 \forall x_2 \exists y_1 \exists y_2 \ (z \vee y_1 \vee y_2) \wedge (\neg x_1 \vee \neg y_1 \vee y_2) \wedge (x_2 \vee \neg y_2) \wedge (y_1 \vee \neg y_2) \wedge (\neg z \vee \neg y_1)$.
Then $\Phi(z) \approx \forall x_1 \forall x_2 \exists y_1 \ (z \vee x_2 \vee y_1) \wedge (\neg x_1 \vee x_2 \vee \neg y_1) \wedge (z \vee y_1) \wedge (\neg z \vee \neg y_1)$. Here, $\exists y_2$ is eliminated by resolving all clauses over positive y_2 with all clauses over $\neg y_2$. All the resolvents are then added to the matrix, and all clauses over y_2 and $\neg y_2$ are dropped. Notice that in the resulting formula, the third clause subsumes the first clause, so we obtain $\Phi(z) \approx \forall x_1 \forall x_2 \exists y_1 \ (\neg x_1 \vee x_2 \vee \neg y_1) \wedge (z \vee y_1) \wedge (\neg z \vee \neg y_1)$ after simplification. If we continue analogously on $\exists y_1$, we finally end up with $\Phi(z) \approx z$.

Proposition 2.5.2. *Let Φ be a QCNF* formula with free variables $\mathbf{z}$ and an existential quantifier $\exists y$ in the innermost quantifier block. Without loss of generality, $\Phi = Q\phi = Q_1 v_1 ... Q_n v_n \exists y \ \phi$. Let $\phi_y := \{C \mid C \in \phi, \ y \in C \text{ or } \neg y \in C\}$ be the subset of clauses with y or $\neg y$, and analogously, $\phi_{\not y} := \{C \mid C \in \phi, \ y, \neg y \notin C\}$. Furthermore, $R_y(\Phi) := \left\{ \sigma \,\middle|\, \exists y \ \phi_y \mid\!\!\overset{*}{\underset{Q\text{-}Res}{}}\ \sigma, \ \sigma \text{ a clause over } \mathbf{z} \cup \{v_1, ..., v_n\} \right\}$. Then $\Phi \approx Q_1 v_1 ... Q_n v_n \ R_y(\Phi) \wedge \phi_{\not y}$.*

A proof follows immediately from Theorem 7.4.6 in [KBL99] if we apply it on the subformula $\exists y \ \phi_y$ and treat $\mathbf{z} \cup \{v_1, ..., v_n\}$ as free variables. Notice that it is sufficient to resolve only on y and not on the free variables. This can be justified by the argument that such resolvents could be removed from R_y retroactively by applying backwards the well-known property that $\phi \approx \phi \wedge \sigma$ for any propositional resolvent σ of a propositional formula ϕ.

In practice, this technique is problematic due to the rapid formula growth from having to generate a potentially quadratic number of resolvents. In addition, it usually leads to a sharp and quick increase in the average clause length of a formula, which is a serious problem for most current *QBF** solvers. Accordingly, we will focus in the following on alternative quantifier elimination techniques that are easier to handle, but we will come back to this approach in Section 5.6 for a discussion on how to integrate it with the other methods.

2.6. Expressive Power of Quantified Boolean Formulas

In Section 1.1, *QBF** formulas have been motivated as a means to provide (potentially) shorter equivalent representations of propositional formulas. For a given propositional formula $\phi(z_1,...,z_r)$ over variables $\mathbf{z} = z_1,...,z_r$, we can provide a *QBF** formula Ψ with free variables $\mathbf{z} = z_1,...,z_r$ such that both are equivalent in the sense that $\phi(t(z_1),...,t(z_r)) = \Psi(t(z_1),...,t(z_r))$ for every truth value assignment $t(\mathbf{z}) := (t(z_1),...,t(z_r)) \in \{0,1\}^r$ to the (free) variables $\mathbf{z}$. Inside of Ψ, additional quantified variables can be introduced. They are local to Ψ and are not explicitly considered when checking for equivalence. This is a powerful advantage of quantified Boolean formulas over propositional formulas, where equivalence is usually lost when additional variables are introduced.

We have already seen in the introduction that existentially quantified variables can be used as abbreviations for repeating parts in the original formula:

$$\begin{aligned} & (A \vee \neg B \vee C \vee D) \wedge (A \vee \neg B \vee C \vee \neg E) \wedge (A \vee \neg B \vee C \vee F) \\ \approx \quad & \exists y\, (\neg y \vee A \vee \neg B \vee C) \wedge (y \vee D) \wedge (y \vee \neg E) \wedge (y \vee F) \end{aligned}$$

The equivalence is easy to verify by considering all possible resolutions on y as in the previous section. This technique can be extended to situations where each subformula in a given set is combined with each of a second set. For example, let $\phi_1(\mathbf{z}),...,\phi_k(\mathbf{z})$ and $\psi_1(\mathbf{z}),...,\psi_l(\mathbf{z})$ be disjunctions of literals over variables in $\mathbf{z}$. Then a Cartesian product of the form $\bigwedge_{i,j}(\phi_i(\mathbf{z}) \vee \psi_j(\mathbf{z}))$ can be abbreviated as follows:

$$\begin{aligned} & \bigwedge_{\substack{i=1,...,k,\\ j=1,...,l}} (\phi_i(\mathbf{z}) \vee \psi_j(\mathbf{z})) \\ \approx \quad & \exists y \bigwedge_{i=1..k} (\neg y \vee \phi_i(\mathbf{z})) \wedge \bigwedge_{j=1..l} (y \vee \psi_j(\mathbf{z})) \end{aligned}$$

If ϕ_i and ψ_j are long and/or k and l are large, the resulting quantified formula is considerably shorter than the propositional version. By applying such abbreviations repeatedly within a given formula, even better compression can be achieved.

We can even handle singular irregularities within such regular structures. For example, assume that the above combination contains no clauses $(\phi_i \vee \psi_i)$ (for

simplicity, we let both i and j range from 1 to k). With the addition of complementary universal variables, it is possible to block exactly these combinations:

$$\begin{aligned} & \bigwedge_{\substack{i,j=1..k,\\ i\neq j}} (\phi_i(\mathbf{z}) \vee \psi_j(\mathbf{z})) \\ \approx \quad & \forall x_1 ... \forall x_k \exists y \bigwedge_{i=1..k} ((\neg y \vee \phi_i(\mathbf{z}) \vee x_i) \wedge (y \vee \psi_i(\mathbf{z}) \vee \neg x_i)) \end{aligned}$$

The equivalence is again seen by considering all possible resolutions on y. Then the clauses $(\neg y \vee \phi_i(\mathbf{z}) \vee x_i)$ and $(y \vee \psi_i(\mathbf{z}) \vee \neg x_i)$ have complementary universals. According to the definition of Q-resolution, such clauses cannot be resolved and are therefore blocked from the set of resolvents. On the other hand, we have non-complementary universals in the clauses $(\neg y \vee \phi_i \vee x_i)$ and $(y \vee \psi_j \vee \neg x_j)$ whenever $i \neq j$. We can resolve those clauses into $(\phi_i \vee x_i \vee \psi_j \vee \neg x_j)$ and afterwards remove the universals through universal reduction, which produces the original clause $(\phi_i \vee \psi_j)$.

Another technique that we can use to obtain concise *QBF** encodings is the elimination of multiple instantiations of (sub-)formulas with different arguments. Consider the example

$$\varphi = \phi(a_{1,1}, ..., a_{1,l}) \wedge \phi(a_{2,1}, ..., a_{2,l}) \wedge ... \wedge \phi(a_{k,1}, ..., a_{k,l})$$

where $\phi(v_1, ..., v_l)$ is a propositional formula over $v_1, ..., v_l$ which is instantiated multiple times for different arguments $a_{i,j} \in A$. If ϕ is large and/or the number k of copies is huge, the formula can be significantly compressed by working only with one single instantiation of ϕ over quantified variables and then associating the $a_{i,j}$ with those quantified variables, as in the BMC encoding by [DHK05]:

$$\varphi \approx \forall x_1 ... \forall x_l \left(\bigvee_{i=1}^{k} \bigwedge_{j=1}^{l} (x_j \leftrightarrow a_{i,j}) \right) \rightarrow \phi(x_1, ..., x_l)$$

When we combine this idea with the previous suggestions for abbreviating repeating subformulas, we can also handle subformulas which are repeated with renamed variables.

With these compression techniques, *QBF** representations are often much more compact than propositional formulas. In fact, a well-known result is that there exist *QCNF** formulas for which every equivalent propositional *CNF* formula

is exponentially longer [KBL99]. This shows that quantification is indeed a powerful language feature. We assess the expressive power of different classes of formulas by the following relation:

Definition 2.6.1. *(Relations between Formula Classes)*
Let $\mathscr{C}_1$ and $\mathscr{C}_2$ be two classes of propositional or quantified Boolean formulas with free variables. Then we define $\mathscr{C}_1 \leq_{poly-length} \mathscr{C}_2$ if and only if there is a polynomial p such that for all formulas $\Phi \in \mathscr{C}_1$, there exists $\Psi \in \mathscr{C}_2$ with $\Phi \approx \Psi$ and $|\Psi| \leq p(|\Phi|)$. That means every Boolean function which can be represented by a formula in $\mathscr{C}_1$ can also be described by an at most polynomially longer equivalent formula in $\mathscr{C}_2$.

If there is a polynomial q such that we can deterministically compute for all $\Phi \in \mathscr{C}_1$ an equivalent $\Psi \in \mathscr{C}_2$ in time $T(|\Phi|) \leq q(|\Phi|)$, we let $\mathscr{C}_1 \leq_{poly-time} \mathscr{C}_2$.

As usual, we let $\mathscr{C}_1 =_{poly-length} \mathscr{C}_2$ if $\mathscr{C}_1 \leq_{poly-length} \mathscr{C}_2$ and $\mathscr{C}_2 \leq_{poly-length} \mathscr{C}_1$. And we define $\mathscr{C}_1 <_{poly-length} \mathscr{C}_2$ if $\mathscr{C}_1 \leq_{poly-length} \mathscr{C}_2$, but $\mathscr{C}_2 \not\leq_{poly-length} \mathscr{C}_1$. Relations $=_{poly-time}$ and $<_{poly-time}$ are defined analogously.

The remark in the previous paragraph implies that $CNF <_{poly-length} QCNF^*$. We will refine this relationship in Chapter 3 by considering classes of quantified Horn formulas, and we will also discuss the role of conjunctive normal form transformations in this context.

2.7. Boolean Function Models

A suitable concept for describing the satisfying truth assignments to the existential variables is the notion of *models* for quantified Boolean formulas, which has initially been introduced in [KBSZ04]. We also use the term *Boolean function model* to distinguish this concept from other kinds of models in different language domains whenever the distinction is not clear from the context. A Boolean function model maps each existential variable y_i to a Boolean function f_{y_i} over universal variables whose quantifiers precede the quantifier of y_i and over free variables (if they are allowed). The *model functions* f_{y_i} can be given in different representations: these might be truth tables, propositional formulas, or even QBF^* formulas themselves. In the following, we represent them as propositional formulas.

We first consider models for closed formulas and then discuss a suitable extension to formulas with free variables. In the closed case, a model is called a *satisfiability model* if substituting the model functions for the existential variables leads to a formula which is true. Consider a two-player game represented by the *QBF* formula

$$\Phi = \forall x_1 \exists y_1 ... \forall x_n \exists y_n \, G(x_1, y_1, ..., x_n, y_n)$$

where x_i is the i-th move of the first player and y_j is the j-th move of the second player. The moves are binary, and the function G determines for a given sequence $x_1, y_1, ..., x_n, y_n$ of moves which player wins. A model indicates which moves y_i the second player makes depending on the preceding moves $x_1, ..., x_i$ of player 1. If we let $G = 1$ whenever player 2 wins, a satisfiability model describes a winning strategy for player 2, which means that for any sequence of opponent moves $x_1, ..., x_i$, he can find suitable moves y_i such that finally $G(x_1, y_1, ..., x_n, y_n) = 1$.

Satisfiability models are essential for our work, so we provide a formal definition (based on [KBSZ04]):

Definition 2.7.1. *(Satisfiability Model)*
For $\Phi \in QBF$ with existential variables $\mathbf{y} = (y_1, ..., y_m)$, let $M = (f_{y_1}, ..., f_{y_m})$ be a mapping which associates with each existential variable y_i a propositional formula f_{y_i} over universal variables whose quantifiers precede the quantifier of y_i. Then M is a ***satisfiability model*** *for Φ if and only if the resulting formula $\Phi[\mathbf{y}/M] := \Phi[y_1/f_{y_1}, ..., y_m/f_{y_m}]$, where simultaneously each existential variable y_i is replaced by its corresponding formula f_{y_i} and the existential quantifiers are dropped from the prefix, is true.*

As introduced in [KBZ05], *equivalence models* extend the notion of models to formulas with free variables by allowing that the model functions f_{y_i} may also depend on free variables. Instead of asking that $\Phi[\mathbf{y}/M]$ must be satisfiable, equivalence models require that Φ and $\Phi[\mathbf{y}/M]$ are equivalent. That makes the concept fit nicely with the main application of providing short equivalent representations of propositional formulas.

Definition 2.7.2. *(Equivalence Model)*
Let $\Phi(\mathbf{z}) = Q\,\phi(\mathbf{x},\mathbf{y},\mathbf{z})$ be a QBF formula with prefix Q and matrix ϕ, universal variables $\mathbf{x} = (x_1,...,x_n)$, existential variables $\mathbf{y} = (y_1,...,y_m)$ and free variables $\mathbf{z} = (z_1,...,z_r)$. For propositional formulas f_{y_i} over $\mathbf{z}$ and over universal variables whose quantifiers precede $\exists y_i$, we say $M = (f_{y_1},...,f_{y_m})$ is an **equivalence model** for $\Phi(\mathbf{z})$ if and only if $\Phi(\mathbf{z}) \approx \forall x_1 ... \forall x_n\, \phi(x_1,...,x_n,\mathbf{y},\mathbf{z})[\mathbf{y}/M]$.*

Models are of particular interest, because they provide a precise characterization of the behavior of the existentially quantified variables. Knowing (parts of) the model can be helpful when solving a quantified Boolean formula. For example, a solver can use models as a formalism to represent information about the existentials which it has learned while traversing the search space. A popular example of this technique is the *QBF* solver sKizzo [Ben05a], which is based on successively computing the model functions and storing them compactly as reduced ordered binary decision diagrams (ROBDDs). In the introduction, we have already pointed out that there appears to be a close connection between the structure of formulas and the structure of their models. This is not only helpful for investigating the expressiveness of particular formula classes. We also want to use information about the model structure to simplify formulas. The following chapter shows how the Horn property affects satisfiability and equivalence models.

3. Quantified Horn Formulas: Models and Transformations

In this chapter, quantified Horn formulas are investigated. We prove that the behavior of the existential quantifiers depends only on the cases where at most one of the universally quantified variables is zero. Accordingly, we give a detailed characterization of *QHORN* satisfiability models which describe the set of satisfying truth assignments to the existential variables. The result is extended to quantified Horn formulas with free variables (*QHORN**) by showing that they have monotone equivalence models.

The main application of these findings is that the general method of universal expansion can be refined such that arbitrary quantified Horn formulas can always be rewritten into short existentially quantified Horn formulas. We prove: a quantified Horn formula Φ of length $|\Phi|$ with free variables, $|\forall|$ universal quantifiers and an arbitrary number of existentials can be transformed in time $O(|\forall| \cdot |\Phi|)$ into an equivalent Horn formula of length $O(|\forall| \cdot |\Phi|)$ which has only existential quantifiers.

We obtain a new algorithm for solving the satisfiability problem for quantified Horn formulas with or without free variables in time $O(|\forall| \cdot |\Phi|)$ by transforming the input into a satisfiability-equivalent propositional formula. In addition, it is shown that *QHORN* satisfiability models can be found with the same complexity.

Our main results can also be generalized to quantified Horn formulas with the extension of allowing an arbitrary number of positive free literals per clause (*QHORN*^{b}). It follows that *QHORN*^{b} satisfiability is *NP*-complete. Interesting applications that we discuss are the Tseitin *CNF* transformation and the encoding of Boolean circuits over $\{\wedge, \vee, \neg\}$. We present *propositionally-labeled series-parallel graphs*, a class of directed acyclic multigraphs which can be encoded as $\exists$*2-HORN*$^b \subseteq$ *QHORN*^{b} formulas. That leads to a novel structure-preserving transformation of propositional formulas into satisfiability-equivalent *3-CNF* of linear length.

We also consider random *QHORN* formulas and present a suitable generation model which avoids trivial unsatisfiability revealed in existing approaches. Experiments illustrate typical sat-unsat-transition behavior and rather simple satisfiability models with characteristic distribution patterns.

This chapter extends preliminary results which have been published in [BKBZ05, BKB08, KBZB09]. The main results have also been pointed out in [KBB09]. Sections 3.8.2 and 3.8.3 on graph encodings and *CNF* transformation are a refinement of initial results from [BKB09].

3.1. Motivation

We want to begin our investigations on the expressiveness of quantifiers by considering quantified Horn formulas. This subclass is an ideal research object, because it appears to be much easier than arbitrary $QCNF^*$, while at the same time being sufficiently powerful for some interesting applications.

The class of quantified Horn formulas ($QHORN$ or $QHORN^*$ if free variables are allowed) contains all quantified Boolean formulas in *CNF* whose clauses have at most one positive literal. As mentioned in Section 2.3, this is one of the tractable QBF^* subclasses. $QHORN^*$-SAT is known to be decidable in time $O(|\forall| \cdot |\Phi|)$ for formulas of length $|\Phi|$ and with $|\forall|$ universal quantifiers [FKKB95].

Horn clauses such as $C = y_0 \vee \neg y_1 \vee ... \vee \neg y_l$ can be thought of as implications $(y_1 \wedge ... \wedge y_l) \rightarrow y_0$ where the premise is a conjunction of positive literals and the conclusion is (at most) one positive literal. Being able to represent this simple version of the "`if-then`" statement in a tractable subclass of QBF^* is part of the importance of the class $QHORN^*$.

Another important point is that $QHORN^*$ formulas may occur as subproblems when solving arbitrary $QCNF^*$ formulas. In fact, it has been shown [CMLBL05] that it can actually be rewarding for DPLL-style backtracking solvers to apply the splitting rule in such a way that Horn (or Horn-renamable) branches are reached as quickly as possible.

There is also an interesting relationship between quantified Horn formulas and the Tseitin procedure [Tse70], which is used to transform arbitrary propositional

formulas in *NNF* into short satisfiability-equivalent *CNF* formulas by introducing abbreviations for misplaced conjunctions. The quantified version of the algorithm produces existentially quantified $QCNF^*$ formulas where the free variables are the original variables of the propositional input formula, and the existential variables are helper variables that introduce abbreviations. It can be shown that the existential variables alone constitute a Horn formula. This motivates us to consider a generalization of $QHORN^*$ in which the Horn property is only enforced on the quantified variables, allowing an arbitrary number of free literals with arbitrary polarity in each clause.

Definition 3.1.1. *(Generalized Quantified Horn Formulas $QHORN^b$)*
With $QHORN^b$, we denote the class of formulas of the form

$$\Phi(z_1,...,z_r) = Q_1 v_1 ... Q_k v_k \; \phi(v_1,...,v_k,z_1,...,z_r) \in QCNF^*$$

such that $\phi \in HORN$ after removing all literals over free variables.

With this powerful extension, we can no longer expect tractability, since this class obviously includes propositional *CNF*. But we will see in Section 3.8 that some positive properties of ordinary $QHORN^*$ formulas remain the same for $QHORN^b$. We will also present further applications, such as encodings of graphs or Boolean circuits.

3.2. Research Goals and Related Work

The goal of this chapter is to investigate whether we can simplify arbitrary $QHORN^*$ formulas by dropping or substituting quantified variables. This is linked to the fundamental question if the syntactic restriction of having at most one positive literal per clause affects the semantics of those formulas in such a way that we can transform them into simplified equivalent formulas. It is known that there exist $QHORN^*$ formulas for which every equivalent propositional Horn or even arbitrary *CNF* formula is exponentially longer [KBL99], which means $CNF <_{poly-length} QHORN^*$. This marks the limit of simplifying $QHORN^*$ formulas and shows that these formulas are actually more complex than their tractability might suggest.

Is it possible to avoid the exponential growth from transforming a quantified formula into a purely propositional Horn formula by eliminating just one kind of

quantifier? In [KBL99], the impossibility of a polynomial-size transformation from *QHORN** to *HORN* is proven by considering a counterexample which is only an existentially quantified Horn formula. That means we know that existential quantifiers can cause exponential blowup, but what about the universal ones? In the introduction, we have already assumed that in *QCNF** formulas, the universal quantifiers are slightly less powerful, but their elimination is clearly exponential in general. It is known that *Q2-CNF** formulas are equivalent to existentially quantified *2-CNF* formulas of linear length [KBL99], but *Q2-CNF** appears to be a simpler class of formulas than *QHORN**, not to mention *QHORN*^{b}. In particular, a forall-reduced *Q2-CNF** formula may only contain at most one universal literal per clause, and only in combination with an existential literal which depends on that universal. The Horn property, on the other hand, allows multiple universals per clause, and also in combination with existentials that do not depend on them. Can we nevertheless transform a *QHORN** or even *QHORN*^{b} formula with arbitrary alternations of existential and universal quantifiers into a purely existentially quantified formula of polynomial length?

For second-order Horn logic, [Grä92] has shown that in a formula of the form $\forall P \exists Q_1 ... \exists Q_r \forall \mathbf{z}\ \phi$, where ϕ is a conjunction of Horn clauses, it is sufficient to consider only predicates P which are false in at most one point. This allows expanding the universal quantifier by considering one instance of the formula for the predicate being always true and one for the predicate being false in exactly one point. That is similar to the well-known universal expansion in *QBF** [AB02, Bie05], where $\forall x\ \phi(x) \approx \phi(0) \wedge \phi(1)$ as mentioned in the introductory chapter. For formulas with multiple existential quantifiers, [Grä92] suggests repeated expansion of the innermost universal quantifier, but that may obviously lead to exponential formula growth. Similarly, existing *QBF** applications [AB02, Bie05] are also exponential in nature.

Our idea is to extend the observation made in [Grä92] to multiple quantifiers by considering not only one single universal quantifier in isolation, but to expand all of them at the same time. Then we show that only those cases are relevant in which at most one of the universally quantified variables is false. Please notice the subtle, but very significant, difference between this statement and the one made by Grädel about one single universally quantified predicate being false in at most one point. For the easy special case of quantified Horn formulas with $\forall^* \exists^*$ ($= \Pi_2$) prefix, that means without complicated quantifier dependencies, the property that we suggest has already been shown in [KKBS87], but without making the important connection to quantifier expansion.

Overall, we extend the existing results with three main contributions:

1. We prove that our observations apply to $QHORN^*$ formulas with *arbitrary* prefix. And while [KKBS87] has been based on Q-resolution, a powerful but rather indirect approach, we will trace the property back to basic results on propositional Horn formulas. This will later allow us to generalize our results even further:

   a) to the powerful class $QHORN^b$ which allows an arbitrary number of free literals with arbitrary polarity in each clause (see Section 3.8).

   b) to Horn formulas with partially ordered quantifiers (see Section 4.7).

2. Our second main contribution is that we do not only investigate the satisfiability decision problem and the connection to the universal variables, but we also provide a thorough characterization of the satisfying truth assignments that are being made to the existential variables.

3. The most important contribution is that we link these results to the well-known universal quantifier expansion algorithm that was introduced in Section 1.2. Unlike existing transformations [Grä92, AB02, Bie05] which are exponential in the worst case, we show that universal expansion can be restricted for $QHORN^*$ to remain quadratic in size and time. This allows us to efficiently transform such formulas into short equivalent purely existentially quantified Horn formulas.

For our investigations, we need a way to characterize the behavior of the quantified variables. In Section 2.7, Boolean function models have been introduced as a suitable tool for describing the satisfying truth assignments to the existential variables depending on the values of the universals (and the free variables, if the formula is not closed). We are in particular interested in making a connection between properties of the model structure and possible formula simplifications. But as motivated in Section 2.7, Boolean function models are also an important topic of their own, so we keep the following discussion on models more general.

The interest in models for quantified Boolean formulas has also sparked some previous work on models for quantified Horn formulas. In [KBSZ04], it has been shown that *QHORN* formulas without free variables have satisfiability models consisting of K_2 functions, which have the form $f_y(x_1,...,x_n) = \bigwedge_{i\in I} x_i$ (or the constants $f_y = 0$ resp. $f_y = 1$). Moreover, [CD05] has made the unproven observation that such K_2 model functions can be further decomposed into individual

strategies where all but one universals are fixed. We present and prove a more general result that also applies to the most interesting case of formulas with free variables by making the connection to universal expansion. In addition, we make the important refinement that we can also specify explicitly the individual strategies that form a model function, which allows us to remove redundancy and combine everything into concise closed-form model functions given as propositional formulas. That enables us to prove a detailed characterization of *QHORN* satisfiability models which encompasses the previous results on K_2 models as direct consequences.

Our precise description of the models makes it possible to compute them surprisingly efficiently and answers the open question raised in [KBSZ04] whether *QHORN* satisfiability models could be computed with the same complexity as the best known satisfiability algorithms for *QHORN*.

Another open problem on models for quantified Horn formulas has been identified in [KBZ05]: how can we generalize the results on satisfiability models for closed *QHORN* formulas to equivalence models for *QHORN** formulas with free variables? As explained in Section 2.7, equivalence models must lead to a tautology for each satisfying assignment to the free variables. That appears to be a more challenging requirement which subsequently calls for more complex models. In fact, we can show that K_2 functions are not even sufficient as equivalence models for the restricted class of purely existentially quantified Horn formulas. So, do we need to go from model functions with conjunctions of positive variables all the way to arbitrary propositional formulas? Fortunately, this is not the case. We prove that negation is still not needed, and monotone functions with conjunctions and disjunctions of positive universals and free variables are powerful enough to express equivalence models for arbitrary *QHORN** formulas.

We want to complement our theoretical results with empiric investigations of particular formula instances. For this purpose, we have identified random formulas to be a suitable research object, because it is easy to create large sets of instances with desired parameters. Furthermore, appropriate generation models for random formulas are an interesting topic on their own, and apparently no previous work on random quantified Horn formulas has been undertaken.

3.3. Satisfiability Models for *QHORN* Formulas

We begin our investigations with examining the behavior of the quantified variables using Boolean function models. In this section and in 3.4, we consider closed *QHORN* formulas. Section 3.5 will then discuss how our observations can be extended to $QHORN^*$ formulas with free variables.

3.3.1. Partial Satisfiability Models

As motivated before, the number of zeros assigned to the universals appears to be an important criterion for our investigations, so we first introduce some useful notation.

Definition 3.3.1. *By B_n^i, we denote the bit vector of length n where only the i-th element is zero, i.e. $B_n^i := (b_1, ..., b_n)$ with $b_i = 0$ and $b_j = 1$ for $j \neq i$.*

Moreover, we define the following relations on n-tuples of truth values:

1. $Z_{\leq 1}(n) = \bigcup_i \{B_n^i\} \cup \{(1, ..., 1)\}$ *(at most one zero)*
2. $Z_{=1}(n) = \bigcup_i \{B_n^i\}$ *(exactly one zero)*
3. $Z_{\geq 1}(n) = \{(a_1, ..., a_n) \mid \exists i : a_i = 0\}$ *(at least one zero)*

For example, if $n = 3$, we have the following relations:

$$\begin{aligned} Z_{\leq 1}(3) &= \{(0,1,1),(1,0,1),(1,1,0),(1,1,1)\} \\ Z_{=1}(3) &= \{(0,1,1),(1,0,1),(1,1,0)\} \\ Z_{\geq 1}(3) &= \{(0,0,0),(0,0,1),(0,1,0),(0,1,1),(1,0,0),(1,0,1),(1,1,0)\} \end{aligned}$$

We omit the parameter n and simply write $Z_{\leq 1}$ (or $Z_{=1}$ resp. $Z_{\geq 1}$) when it is clear from the context. Usually, n equals the total number of the universal quantifiers in a given formula.

Our goal is to show that for quantified Horn formulas, we do not need to consider all possible truth assignments to the universal variables. We restrict those assignments according to a relation $R_\forall(n) \subseteq \{0,1\}^n$ on the set of possible truth

assignments to n universals. We first introduce the concept for *QBF* formulas in general. Then we apply it to *QHORN* formulas, in which case $R_\forall$ will restrict the number of zeros using the relations given in the previous definition.

Let $\Phi = Q\phi(\mathbf{x},\mathbf{y}) \in QBF$. The definition of a satisfiability model in Section 2.7 requires that substituting the existentials $\mathbf{y}$ in Φ produces a formula $\Phi[\mathbf{y}/M]$ which is true. That means the matrix $\phi[\mathbf{y}/M]$ must be true for all possible assignments to the universals $\mathbf{x}$. We now introduce a special kind of satisfiability model which weakens this condition: a so-called *$R_\forall$-partial satisfiability model* is only required to satisfy $\phi[\mathbf{y}/M]$ for certain truth assignments to the universal variables which are given by a relation $R_\forall$.

Definition 3.3.2. *($R_\forall$-partial Satisfiability Model)*
*For $\Phi = Q\phi(\mathbf{x},\mathbf{y}) \in QBF$ with universal variables $\mathbf{x} = (x_1,...,x_n)$ and existential variables $\mathbf{y} = (y_1,...,y_m)$, let $M = (f_{y_1},...,f_{y_m})$ be a mapping which associates with each existential variable y_i a propositional formula f_{y_i} over universal variables whose quantifiers precede the quantifier of y_i. Furthermore, let $R_\forall(n)$ be a relation on the set of possible truth assignments to the n universals. Then M is a **$R_\forall$-partial satisfiability model** for Φ if the formula $\phi[\mathbf{y}/M]$ is true for all $\mathbf{x} \in R_\forall(n)$.*

For the sake of completeness, we also allow $n = 0$ (i.e. formulas without universal variables) in the above definition, in which case the f_{y_i} are constants 0 or 1, and we require that $\phi[\mathbf{y}/M]$ is true.

Consider the following example: the formula $\Phi = \forall x_1 \forall x_2 \exists y\,(x_1 \vee y) \wedge (x_2 \vee \neg y)$ does not have a satisfiability model, but $M = (f_y)$ with $f_y(x_1,x_2) = \neg x_1 \vee x_2$ is a $Z_{\leq 1}$-partial satisfiability model for Φ because of $\phi[y/M] = (x_1 \vee \neg x_1 \vee x_2) \wedge (x_2 \vee (x_1 \wedge \neg x_2)) \approx x_2 \vee x_1$, which is true for all $\mathbf{x} = (x_1,x_2)$ with $\mathbf{x} \in Z_{\leq 1}$.

3.3.2. The Core of *QHORN* Satisfiability Models

It is not surprising that the mere existence of a $Z_{\leq 1}$-partial satisfiability model does not imply the existence of a (total) satisfiability model - at least not in the general case. Interestingly, this implication is indeed true for quantified Horn formulas. We now show: if we can find a $Z_{\leq 1}$-partial satisfiability model M to satisfy a quantified Horn formula whenever at most one of the universals is

false, then we can also satisfy the formula for arbitrary truth assignments to the universals.

Even better, we can precisely characterize the satisfying truth assignment to the existential variables for arbitrary values of the universals by deriving it from the behavior of the existentials whenever at most one universal is false. The partial model M is the relevant core which contains all information that is needed to construct a total satisfiability model M^t. The following definition provides the details of that construction.

Definition 3.3.3. *(Total Completion of Partial Models)*
Let $\Phi = Q\phi(\mathbf{x},\mathbf{y}) \in$ QHORN be a quantified Horn formula with universal variables $\mathbf{x} = (x_1,...,x_n)$ and existentials $\mathbf{y} = (y_1,...,y_m)$, and let $M = (f_{y_1},..,f_{y_m})$ be a $Z_{\leq 1}$-partial satisfiability model for Φ. For each $f_{y_i}(x_1,...,x_{n_i})$ in M, we define $f^t_{y_i}$ as follows:

$$\begin{aligned} f^t_{y_i}(x_1,...,x_{n_i}) \quad := \quad & (x_1 \ \vee f_{y_i}(0,1,1,...,1)) \\ \wedge \quad & (x_2 \ \vee f_{y_i}(1,0,1,...,1)) \\ \wedge \quad & ... \\ \wedge \quad & (x_{n_i} \vee f_{y_i}(1,1,...,1,0)) \\ \wedge \quad & f_{y_i}(1,...,1) \end{aligned}$$

*Then we call $M^t = (f^t_{y_1},...,f^t_{y_m})$ the **total completion** of M.*

Please notice that the previous definition is equivalent to the following:

$$\begin{aligned} f^t_{y_i}(x_1,...,x_{n_i}) \quad = \quad & (\neg x_1 \ \rightarrow f_{y_i}(0,1,1,...,1)) \\ \wedge \quad & (\neg x_2 \ \rightarrow f_{y_i}(1,0,1,...,1)) \\ \wedge \quad & ... \\ \wedge \quad & (\neg x_{n_i} \rightarrow f_{y_i}(1,1,...,1,0)) \\ \wedge \quad & f_{y_i}(1,...,1) \end{aligned}$$

This means that when some of the arguments are zero, we consider *all cases* where *at most one* of those arguments is zero and take the conjunction of the corresponding original function values. For example, $f^t_y(1,0,0,1) = f_y(1,0,1,1) \wedge f_y(1,1,0,1) \wedge f_y(1,1,1,1)$. In case all the arguments are 1, we simply return the value of the original function, i.e. $f^t_y(1,..,1) = f_y(1,..,1)$. These observations lead to the following lemma.

Lemma 3.3.4. *Let $t(\mathbf{x}) = (t(x_1),...,t(x_n)) \in Z_{\geq 1}(n)$, $t(x_{z_1}) = 0,...,t(x_{z_k}) = 0$ and $t(x_s) = 1$ for $s \neq z_1,...,z_k$, be a truth assignment to the universal variables where $k \geq 1$ universals $x_{z_1},...,x_{z_k}$ are zero. Then the definition of $f^t_{y_i}$ implies*

$$f^t_{y_i}(t(x_1),...,t(x_{n_i})) = \bigwedge_{1\leq j\leq k} f_{y_i}(t_{z_j}(x_1),...,t_{z_j}(x_{n_i})) \wedge f_{y_i}(1,...,1)$$

where $t_{z_j}(\mathbf{x}) = (t_{z_j}(x_1),...,t_{z_j}(x_n)) = B^{z_j}_n$ is a truth assignment where exactly one universal x_{z_j} is zero.

Moreover, total completion equals the partial model when all universals on which y_i depends are 1:

$$f^t_{y_i}(1,...,1) = f_{y_i}(1,...,1)$$

This definition is based on an observation: it is a well-known fact about *propositional* Horn formulas, proved by Alfred Horn himself [Hor51], that the intersection of two satisfying truth assignments is a satisfying truth assignment, too. Let $t_1(\mathbf{x}) = (t_1(x_1),...,t_1(x_n)) \in \{0,1\}^n$ and $t_2(\mathbf{x}) = (t_2(x_1),...,t_2(x_n)) \in \{0,1\}^n$ be two truth assignments over variables $x_1,...,x_n$, then the intersection of t_1 and t_2 is defined as

$$t_1(\mathbf{x}) \cap t_2(\mathbf{x}) = (t_1(x_1) \wedge t_2(x_1),...,t_1(x_n) \wedge t_2(x_n)) \ .$$

Our idea is to establish a similar relationship between the satisfying truth assignments to the existential variables in a quantified Horn formula, taking also into consideration the universally quantified variables. Assume that a *QHORN* formula with two universal variables x_i and x_j is known to be satisfiable when $x_i = 0$ and $x_j = 1$ or when $x_i = 1$ and $x_j = 0$. That means there exist two truth assignments t_1 and t_2 to the existential variables such that the formula is satisfied in both cases. If we lift the closure under intersection to the quantified case, it means that the intersection of t_1 and t_2 satisfies the formula when both x_i and x_j are zero.

An important point to consider is that we have to obey the quantifier dependencies when choosing truth values for the existential variables. Assume the previous example includes an existential variable y_k with $t_1(y_k) = 1$ and $t_2(y_k) = 0$ and the additional restriction that $\exists y_k$ occurs earlier in the prefix than $\forall x_j$. Then y_k does not depend on x_j, but the intersection of t_1 and t_2 would assign y_k the

value 0 when $x_i = 0$ and $x_j = 0$, which is not allowed, because we have already set y_k to 1 when $x_i = 0$ (but $x_j = 1$). This shows that intersecting arbitrary satisfying truth assignments is not appropriate for *QHORN* formulas. However, the proof of Theorem 3.3.5 guarantees by construction that quantifier dependencies are respected. Another point to notice is that we always intersect with $f_{y_i}(1,...,1)$. This makes sure that we reduce $f^t_{y_i}$ to a well-defined value from the partial satisfiability model in cases where all zeros are assigned to universals on which y_i does not depend.

Theorem 3.3.5. *Let $\Phi = Q\phi(\mathbf{x},\mathbf{y}) \in$ QHORN be a quantified Horn formula which has a $Z_{\leq 1}$-partial satisfiability model $M = (f_{y_1},..,f_{y_m})$. Then its total completion $M^t = (f^t_{y_1},...,f^t_{y_m})$ as defined above is a satisfiability model for Φ.*

Proof:
We must show that $\phi[\mathbf{y}/M^t]$ is true for all truth assignments to the universal variables. Since $f^t_{y_i}(1,...,1) = f_{y_i}(1,...,1)$, we only need to consider truth assignments where at least one universal is zero.

Let $t(\mathbf{x}) = (t(x_1),...,t(x_n)) \in Z_{\geq 1}(n)$ with $t(x_{z_1}) = 0$, ..., $t(x_{z_k}) = 0$ and $t(x_s) = 1$ for $s \neq z_1,...,z_k$ be a truth assignment to the universal variables where $k \geq 1$ universals x_{z_1}, ..., x_{z_k} are zero. When we combine the truth assignment to the universals and the corresponding values of the model functions into a $(n+m)$-tuple of truth values, we obtain the following bit vector:

$$\tau = (t(x_1),...,t(x_n), f^t_{y_1}(t(x_1),...,t(x_{n_1})),...,f^t_{y_m}(t(x_1),...,t(x_{n_m})))$$

Our goal is to prove that the propositional matrix ϕ is true under the truth value assignment $\tau = (\tau(x_1),...,\tau(x_n),\tau(y_1),...,\tau(y_m))$. We can write the tuple $t(\mathbf{x})$ with k universals being zero as an intersection $t(\mathbf{x}) = t_{z_1}(\mathbf{x}) \cap ... \cap t_{z_k}(\mathbf{x})$ of k assignments $t_{z_j}(\mathbf{x}) = B^{z_j}_n$ with exactly one zero each. Similar to the definition of $f^t_{y_i}$, it is useful to intersect with $(1,..,1)$ as well. With this trick, we have $t(\mathbf{x}) = t_{z_1}(\mathbf{x}) \cap ... \cap t_{z_k}(\mathbf{x}) \cap (1,...,1)$ and can decompose τ as follows:

$$\begin{aligned}
\tau = \quad & (t_{z_1}(\mathbf{x}), f^t_{y_1}(t(\mathbf{x}_{1..n_1})),...,f^t_{y_m}(t(\mathbf{x}_{1..n_m}))) \\
\cap \quad & \cdots \\
\cap \quad & (t_{z_k}(\mathbf{x}), f^t_{y_1}(t(\mathbf{x}_{1..n_1})),...,f^t_{y_m}(t(\mathbf{x}_{1..n_m}))) \\
\cap \quad & (1,...,1, f^t_{y_1}(t(\mathbf{x}_{1..n_1})),...,f^t_{y_m}(t(\mathbf{x}_{1..n_m})))
\end{aligned}$$

For clarity, we abbreviate $t(\mathbf{x}_{1..n_i}) := (t(x_1),...,t(x_{n_i}))$ and $f(\mathbf{1}) := f(1,...,1)$.

Now, Lemma 3.3.4 allows us to decompose this even further:

$$\begin{aligned}
\tau = \quad & \left(t_{z_1}(\mathbf{x}), \bigwedge_{j=1..k} f_{y_1}(t_{z_j}(\mathbf{x}_{1..n_1})) \wedge f_{y_1}(\mathbf{1}), ..., \bigwedge_{j=1..k} f_{y_m}(t_{z_j}(\mathbf{x}_{1..n_m})) \wedge f_{y_m}(\mathbf{1})\right) \\
\cap \quad & \cdots \\
\cap \quad & \left(t_{z_k}(\mathbf{x}), \bigwedge_{j=1..k} f_{y_1}(t_{z_j}(\mathbf{x}_{1..n_1})) \wedge f_{y_1}(\mathbf{1}), ..., \bigwedge_{j=1..k} f_{y_m}(t_{z_j}(\mathbf{x}_{1..n_m})) \wedge f_{y_m}(\mathbf{1})\right) \\
\cap \quad & \left(1, ..., 1, \bigwedge_{j=1..k} f_{y_1}(t_{z_j}(\mathbf{x}_{1..n_1})) \wedge f_{y_1}(\mathbf{1}), ..., \bigwedge_{j=1..k} f_{y_m}(t_{z_j}(\mathbf{x}_{1..n_m})) \wedge f_{y_m}(\mathbf{1})\right)
\end{aligned}$$

This can be simplified by distributing the conjunctions over the intersections:

$$\begin{aligned}
\tau = \quad & (t_{z_1}(\mathbf{x}), f_{y_1}(t_{z_1}(\mathbf{x}_{1..n_1})), ..., f_{y_m}(t_{z_1}(\mathbf{x}_{1..n_m}))) \\
\cap \quad & \cdots \\
\cap \quad & \left(t_{z_k}(\mathbf{x}), f_{y_1}(t_{z_k}(\mathbf{x}_{1..n_1}), ..., f_{y_m}(t_{z_k}(\mathbf{x}_{1..n_m}))\right) \\
\cap \quad & (1, ..., 1, f_{y_1}(\mathbf{1}), ..., f_{y_m}(\mathbf{1}))
\end{aligned}$$

We have thus split $\tau = (\tau(x_1), ..., \tau(x_n), \tau(y_1), ..., \tau(y_m))$ into an intersection $\tau = \tau_1 \cap ... \cap \tau_k \cap \tau_0$ of $k+1$ individual truth assignments to the universal and existential variables in ϕ. A close look reveals that each τ_i represents a situation where at most one universal is zero and each existential y_i is chosen as determined by f_{y_i} for that constellation of the universals. Under the assumption that $M = (f_{y_1}, .., f_{y_m})$ is a $Z_{\leq 1}$-partial satisfiability model of Φ, we know that ϕ is true under each of those assignments $\tau_0, ..., \tau_k$. Since ϕ is a propositional Horn formula, the intersection of satisfying truth assignments is again a satisfying truth assignment.

By construction, quantifier dependencies are respected, i.e. an existential cannot obtain a different value when only a universal on which it does not depend changes. To see this, we write τ as $\tau = (\tau_1 \cap \tau_0) \cap (\tau_2 \cap \tau_0) \cap ... \cap (\tau_k \cap \tau_0)$. Intersecting τ_i with τ_0 assigns the existentials as determined by M^t when exactly one universal is zero. And in the outer intersections $(\tau_i \cap \tau_0) \cap (\tau_j \cap \tau_0)$, the truth value of an existential can only change if one of the universals on which it depends changes value as well. That is guaranteed, because the arguments of $f_{y_1}, ..., f_{y_m}$ contain only those universals on which the corresponding existentials depend. □

Using Definition 3.3.3 and Theorem 3.3.5, we can immediately obtain a (total) satisfiability model upon finding a $Z_{\leq 1}$-partial satisfiability model for a quantified Horn formula. This means that the behavior of the existential quantifiers is completely determined by the cases where at most one of the universal variables is false. The cases where more than one of them is assigned false are not relevant for predicting the behavior of the existentials.

3.3.3. Model Structure

An immediate consequence of our results on partial satisfiability models is the fact that *QHORN* formulas have models consisting of functions of the form $f_y(x_1,...,x_n) = \bigwedge_{i\in I} x_i$ (or the constants $f_y = 0$ resp. $f_y = 1$). In accordance with [KBSZ04], the formal definition of this class of models is given below.

Definition 3.3.6. *(K_2 Satisfiability Model)*
With K_2, we denote the following class of Boolean functions:

$$K_2 := \{f \mid \exists I \subseteq \{1,...,n\} : f(x_1,...,x_n) = \bigwedge_{i\in I} x_i, n \geq 1\} \cup \{f \mid f = 0 \text{ or } f = 1\}$$

*Let $M = (f_{y_1},...,f_{y_m})$ be a satisfiability model for a formula $\Phi \in QBF$. Then we call M a **K_2 satisfiability model** for Φ if the model functions f_{y_i} are in K_2 for every $1 \leq i \leq m$.*

Theorem 3.3.7. *Any satisfiable formula $\Phi \in QHORN$ has a K_2 satisfiability model. It can be obtained from a $Z_{\leq 1}$-partial satisfiability model through total completion.*

Proof:
If Φ is satisfiable, it has a $Z_{\leq 1}$-partial satisfiability model $M = (f_{y_1},...,f_{y_m})$. According to Definition 3.3.3 and Theorem 3.3.5, its total completion M^t is a (total) satisfiability model and is composed of functions given by

$$\begin{aligned} f^t_{y_i}(x_1,...,x_{n_i}) \quad := \quad & (x_1 \vee f_{y_i}(0,1,1,...,1)) \\ \wedge \quad & (x_2 \vee f_{y_i}(1,0,1,...,1)) \\ \wedge \quad & ... \\ \wedge \quad & (x_{n_i} \vee f_{y_i}(1,1,...,1,0)) \\ \wedge \quad & f_{y_i}(1,...,1) \end{aligned}$$

Notice that $f_{y_i}(0,1,1,...,1)$, $f_{y_i}(1,0,1,...,1)$, ..., $f_{y_i}(1,...,1)$ are merely Boolean constants in the definition of $f^t_{y_i}$.
That means we actually have functions of the form

$$\begin{aligned} f^t_{y_i}(x_1,...,x_{n_i}) \quad := \quad & (x_1 \ \vee c_1) \\ \wedge \quad & (x_2 \ \vee c_2) \\ \wedge \quad & ... \\ \wedge \quad & (x_{n_i} \vee c_{n_i}) \\ \wedge \quad & c_{n_i+1} \end{aligned}$$

with $c_j = 0$ or $c_j = 1$. Clearly, those functions are in K_2. □

In [KBSZ04], it has already been shown that quantified Horn formulas have K_2 models. However, that proof was significantly longer and required more advanced techniques (Q-pos-unit-resolution). Most importantly, however, it did not lead to an efficient algorithm for finding those K_2 models. As mentioned earlier, it has since been an open question whether it would be possible to find K_2 satisfiability models in time at most $O(|\forall| \cdot |\Phi|)$, the complexity of the best known *QHORN*-SAT decision algorithms. In Section 3.7.2, we will develop such an $O(|\forall| \cdot |\Phi|)$ algorithm on the basis of total completion. By virtue of the previous theorem, it produces K_2 models.

An interesting observation in the proof above is that $c_{n_i+1} = f_{y_i}(1,...,1) = 0$ implies $f^t_{y_i}(x_1,...,x_{n_i}) = 0$. This means that existential variables which are assigned the value 0 when all universals are 1 can keep this satisfying assignment for any other combination of values that the universals may have. It is therefore typical for *QHORN* satisfiability models that they contain a large fraction of constant model functions $f_{y_i} = 0$. We have also verified this experimentally for random *QHORN* formulas in the next section.

3.4. Random *QHORN* Formulas

Now that we have characterized in detail the structure of *QHORN* satisfiability models, an interesting question is how the individual model functions for given instances are distributed within K_2: how many conjunctions do they typically have, and how many of them are just Boolean constants? Such model distributions are of general interest to *QBF* solving, because they are indicators of how

complex the search space is, how strong the dependencies between the variables are, etc. Since those factors have a direct impact on the performance of solvers, they are helpful for assessing the hardness of *QBF* formulas. In this section, we are going to investigate model distributions for random quantified Horn formulas and demonstrate that they possess a characteristic and very simple pattern, which we can explain with our previous results on the model structure.

At first, however, we need a suitable method for generating random *QHORN* instances. Besides our interest in satisfiability models, we will give a detailed coverage of random *QHORN* formulas in general. Random formulas and their generation have attracted a considerable amount of previous work, because interesting phase transitions depending on generation parameters have been observed for various classes of propositional or *QBF* formulas. While propositional Horn formulas have been considered before (e.g. in [DBC01, Ist02]), we are not aware of existing results on *QHORN*.

In literature, procedures for generating random formulas are usually called (generation) models. In order to avoid confusion between satisfiability models and generation models, we will avoid using the term "model" without further distinction, even if it deviates from common names. For example, we will say "FCL2 generation model" throughout this section instead of the commonly used name "FCL2 model".

3.4.1. Random Formula Generation

As described in [CGS97], random *QCNF* formulas are typically generated according to a fixed clause length generation model with a given number k of quantifier blocks, all having the same cardinality n. In order to produce a specified number q of clauses with fixed size h, randomly choose for each clause h distinct variables from the whole prefix and negate each one with probability 0.5. Discard duplicate clauses and purely universal clauses.

Gent and Walsh have identified in [GW99] a problem with this method, the so-called unit flaw: for a given clause size h, a formula is likely to contain complementary $\exists$-unit clauses when $q = \Omega(\sqrt{n})$ due to the birthday paradox (about $\sqrt{365}$ people are necessary to have a 50% chance of matching birthdays). For $n \to \infty$, two such complementary clauses $C_1 = y_i \bigvee_{l \in L_1} x_l^{\varepsilon_l}$ and $C_2 = \neg y_i \bigvee_{l \in L_2} x_l^{\varepsilon_l}$ almost certainly have distinct universals, i.e. $L_1 \cap L_2 = \emptyset$. These clauses can

therefore always be resolved to the empty clause, since there is no blocking by complementary universals. That makes the formula trivially unsatisfiable.

Several modified generation models have been suggested to avoid the unit flaw. While the solutions in [GW99] and [CI05] vary in detail, they have in common that they no longer allow clauses with less than two existential variables. For *QHORN* formulas, however, such generation models are unsuitable, because requiring at least two existentials per clause means that each clause has at least one negative existential, which makes the formula trivially satisfiable when all existentials are false.

Another suggestion to make the original method unit-flawless was made by Cadoli et al. in [CSGG02]. In their FCL2 generation model, an $\exists$-unit clause is only removed and replaced with a new clause if the formula already contains a complementary $\exists$-unit clause with non-blocking universals. That is, the two clauses have no complementary universals at all, or all of them can be forall-reduced, because their quantifiers come after the quantifier of the existential unit variable. For example, assume that the prefix is $\exists y_1 \forall x_1 \exists y_2 \forall x_2 \exists y_3$ and that the clauses $C_1 = (y_2 \vee \neg x_1 \vee \neg x_2)$ and $C_2 = (\neg y_3 \vee x_2)$ have already been generated. Then we would discard an additional clause $(\neg y_2 \vee \neg x_1 \vee x_2)$, because x_2 and $\neg x_2$ can be forall-reduced, and the clause resolves with C_1 to the empty clause. On the other hand, a clause $(y_3 \vee \neg x_1 \vee \neg x_2)$ would be allowed, because it cannot be resolved with C_2 due to blocking universals $x_2, \neg x_2$.

Since this approach also works for Horn formulas, we have adopted it for our experiments. Based on the previous discussion, we obtain the following procedure for generating random *QHORN* formulas:

1. we build a prefix with k quantifier blocks, each having n quantifiers. The innermost block is always existential.
2. we produce q clauses of uniform size h. For each clause, distinct variables are chosen randomly from the whole prefix. Each one is negated with probability 0.5.
3. a clause is discarded and replaced with a new one whenever one of the following is true:
   - if the clause is not a Horn clause
   - if it is purely universal

- if a duplicate clause has already been generated
- if the clause is $\exists$-unit and the formula already contains a complementary $\exists$-unit clause with non-blocking universals.

Notice that in this generation model, most clauses are going to be definite, since the probability of all literals in a clause being negative is only 0.5^h. This matches some existing studies on propositional Horn formulas, e.g. [DBC01, Ist02].

3.4.2. Phase Transition Behavior

A typical phenomenon for various classes of random satisfiability problems is a phase transition from mostly satisfiable to mostly unsatisfiable problems induced by small changes of the *#clauses/#variables* ratio. Around a ratio where the probability of a formula being satisfiable is about 50%, this probability changes significantly (even abruptly in case of a sharp transition) when the *#clauses/#variables* ratio is varied slightly. Experience has shown that formulas near this transition are usually more challenging to solve and therefore more suitable for experiments. Further away from the phase transition, formulas usually exhibit simpler reasons for (un)satisfiability.

We have examined this phase transition behavior for the quantified Horn formulas generated with the above model, which we have implemented within our logic framework ProverBox (see Section 5.7.1). In our experiments, formulas with $\forall\exists$ prefix and various clause sizes were considered. For different numbers of variables, we have varied the *#clauses/#variables* ratio and determined the fraction of unsatisfiable problems. The results are shown in Figures 3.1 and 3.2 for clause sizes $h = 5$ and $h = 8$.

As expected, we can observe the typical phase transition patterns. However, it appears that the location of the transition depends on two factors: the clause size h and the number of variables n. We can explain both phenomenons.

We start with the influence of the clause size, which is easy to understand: For clause size $h = 8$, the phase transition is shifted towards larger q/n ratios in comparison to $h = 5$. The reason is that when the number of clauses and variables is the same, constraints with larger clauses are easier to satisfy. We have also performed experiments with smaller clause sizes like the usual *3-CNF*, but in those cases, the transition is so far to the left that the problems are becoming trivially

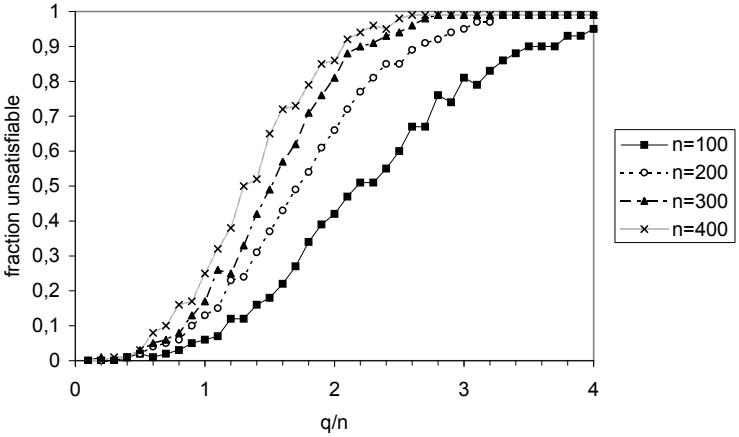

Figure 3.1.: Fraction of unsatisfiable problems for random $\forall\exists$*HORN* formulas with clause size $h = 5$

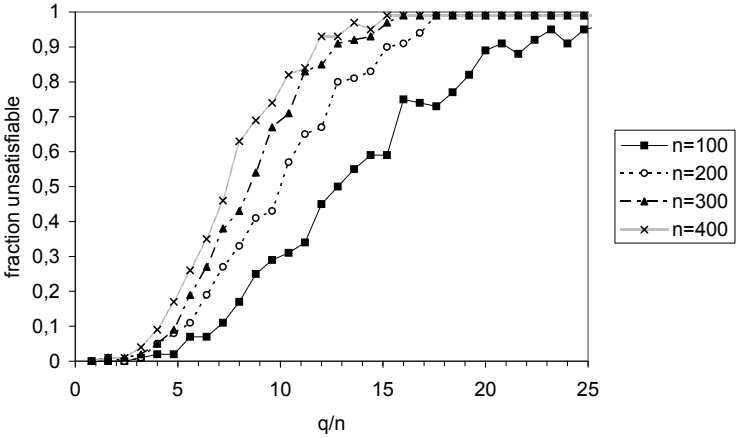

Figure 3.2.: Fraction of unsatisfiable problems for random $\forall\exists$*HORN* formulas with clause size $h = 8$

solvable with standard simplification techniques like pure literal detection.
Even for $h = 5$, this effect has been evident. The reason is that q Horn clauses have at most q positive literals, but we have $2n$ variables. So unless the ratio q/n is significantly greater than 2, we are likely to have some variables which occur only negatively. Removing those usually produces new pure literals, and the formula ultimately collapses.

More surprising is the observation that for a fixed clause size, the phase transition appears to occur earlier for larger n. Interestingly, this is not the case for propositional satisfiability, where the transition appears to occur at a constant q/n ratio [CA93]. On the other hand, the FCL2 generation model by Cadoli et al. [CSGG02] also shows this shift. Since our generation model is based on FCL2, it is not unexpected that our experiments show this behavior as well. Unfortunately, previous literature on FCL2 does not give a reason for this effect. Our explanation is that this behavior is similar to the unit flaw mentioned above, it is just postponed one step: we now have trivial unsatisfiability after two steps of Q-unit resolution. To prove this phenomenon, we need to have a look at the number of $\exists$-unit clauses in formulas produced by the FCL2 generation model.

Lemma 3.4.1. *For fixed clause size h and fixed ratio $\frac{q}{n}$, the FCL2 generation model produces formulas with $\Theta(n) = \Theta(q)$ $\exists$-unit clauses.*

Proof:
For the original fixed clause length generation model in which complementary $\exists$-unit clauses are kept, it is easy to see that the probability of a clause being $\exists$-unit is asymptotically constant: if we draw h variables out of n existentials and n universals without replacement, $\Pr[\text{only 1st variable is existential}] \to \frac{1}{2^h}$ for $n \to \infty$. It follows that $\Pr[\text{exactly one of } h \text{ variables is existential}] \to \frac{h}{2^h}$ for $n \to \infty$.[1]

For the FCL2 generation model, it is also necessary to show that we do not have less than $\Theta(n)$ $\exists$-unit clauses due to the discarding of complementary pairs. Let $u = p \cdot n$ be the number of $\exists$-unit clauses without discarding, where p is asymptotically constant. For any such clause, the probability that it is complementary to another one is $\frac{u}{2n} = \frac{p}{2}$, because that is the probability that a particular $\exists$-unit

[1] If n is not significantly larger than h, we have to consider the effect that after drawing some universal variables, there are more existentials than there are universal variables left. But since this imbalance is bounded by the constant h, it can be neglected as $n \to \infty$.

literal out of the $2n$ possible ones occurs in one of the u $\exists$-unit clauses. Then the expected number of $\exists$-unit clauses for which there exists a complementary $\exists$-unit clause is $\frac{p}{2} \cdot u = \frac{p^2}{2} \cdot n$. In the FCL2 generation model, less than this number of clauses is discarded (only one of a pair of complementary clauses, and only if they do not have blocking universals). Thus, the expected number of $\exists$-unit clauses in the FCL2 model is at least $p \cdot n - \frac{p^2}{2} \cdot n$, which is still $\Theta(n)$. $\square$

We are now able to show that FCL2 produces formulas which can almost certainly be refuted after two steps of Q-unit resolution if n is sufficiently large.

Theorem 3.4.2. *For fixed clause size h and fixed ratio $\frac{q}{n}$, the FCL2 generation model will produce formulas ϕ with*

$$\Pr\left[\phi \mid\!\frac{2}{\text{Q-unit res}} \sqcup\right] \to 1$$

for $n \to \infty$.

Proof:
According to the previous lemma, we expect FCL2 to generate $p_1 \cdot n$ $\exists$-unit clauses, where p_1 is asymptotically constant. Analogously, we expect $p_2 \cdot n$ clauses with exactly two existentials (we call them $\exists$-2 clauses). We know that there are only $2n$ different unit clauses, and there are also only $2n$ different possibilities for the first existential literal in the $\exists$-2 clauses. With probability $1 - \frac{1}{2n}$, the first $\exists$-unit literal is not complementary to the first existential literal in the first $\exists$-2 clause. Then with probability $(1 - \frac{1}{2n})^{2p_2n}$, the first $\exists$-unit literal is not complementary to any existential literal in any $\exists$-2 clause. Therefore, with probability $1 - (1 - \frac{1}{2n})^{2p_2n}$, the first $\exists$-unit literal is complementary to an existential literal in a $\exists$-2 clause. For sufficiently large n, the two complementary clauses will almost certainly have disjoint sets of universals and can therefore be resolved into a new $\exists$-unit clause. Since we have p_1n $\exists$-unit clauses to start with, we will on average obtain

$$u = \left(1 - \left(1 - \frac{1}{2n}\right)^{2p_2n}\right) \cdot p_1n$$

new $\exists$-unit clauses. For $n \to \infty$, we get

$$\lim_{n\to\infty} u = \lim_{n\to\infty} \left(1 - \left(1 - \tfrac{1}{2n}\right)^{2p_2n}\right) \cdot p_1n = \underbrace{\left(1 - e^{-p_2}\right)}_{\Theta(1)} \cdot \underbrace{\lim_{n\to\infty} p_1n}_{\Theta(n)}$$

Now we can use Gent and Walsh's argument about the unit flaw [GW99] on these new $\exists$-units. According to the birthday paradox, we have a 50% chance to find two people with the same birthday in a group of about $\sqrt{365}$ people. In our case, this means that we expect to find complementary literals when we have $\sqrt{2n}$ $\exists$-units. Since $u = \Theta(n)$, there will almost certainly occur complementary $\exists$-units when $n \to \infty$, and they will also most likely have distinct universals, so they can be resolved to the empty clause. □

This trivial unsatisfiability explains why the FCL2 generation model, and also our FCL2-based generation model for random *QHORN* formulas, is showing a shift of the phase transition to the left as $n \to \infty$. We can conclude that unit flaws cannot be avoided by the ad hoc approach of simply disallowing complementary $\exists$-unit clauses with non-blocking universals, because this still allows for trivial unsatisfiability in two steps. It might be possible to refine the approach by disallowing $\exists$-unit clauses which are complementary to $\exists$-2 clauses with non-blocking universals, but this will probably only defer the problem another step.

Since the generation models in [GW99] and [CI05], where $\exists$-units are not allowed at all, do not work for Horn formulas, we need another solution. Rather than not having any $\exists$-units at all, we can reduce them to less than $\Theta(n)$. Theorem 3.4.2 is based on the fact that the number of $\exists$-unit clauses grows linearly, but trivial unsatisfiability occurs at $\Omega(\sqrt{n})$. Our approach is therefore to modify the random formula generation model from Section 3.4.1 such that $\exists$-unit clauses are discarded and replaced with a new clause with probability $1 - 1/\sqrt{n}$, so that we only obtain $\Theta(n \cdot 1/\sqrt{n}) = \Theta(\sqrt{n})$ $\exists$-units. We call this method *FCL2-U*.

Figures 3.3 and 3.4 show the phase transition behavior of the FCL2-U generation model for Horn formulas with clause sizes $h = 5$ and $h = 8$.

We still have a slight shift of the phase transition to the left as n grows, but the effect is much weaker than it was in the original FCL2 model (cf. Figures 3.1 and 3.2). As for the remaining effect (which is actually also weakly observable in [GW99]), we conjecture that with larger n, we have less blocking by complementary universal variables when attempting to resolve two clauses. The fact that this effect is more evident for $h = 8$ than for $h = 5$ appears to confirm this conjecture, because blocking is more frequent for larger clause sizes.

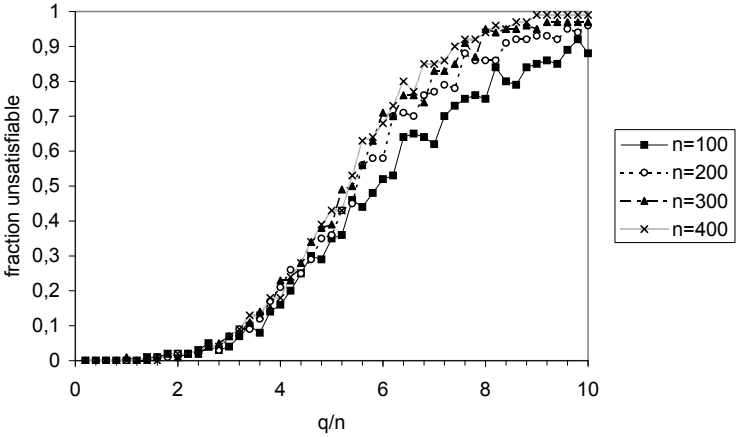

Figure 3.3.: Fraction of unsatisfiable problems for random $\forall\exists$*HORN* formulas with clause size $h = 5$ using the FCL2-U generation model

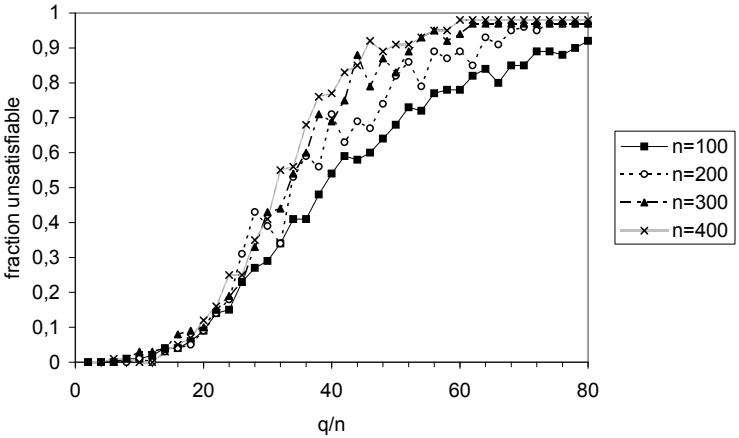

Figure 3.4.: Fraction of unsatisfiable problems for random $\forall\exists$*HORN* formulas with clause size $h = 8$ using the FCL2-U generation model

3.4.3. Satisfiability Model Distributions

Now that we have a suitable generation model for random *QHORN* formulas, we can investigate what the satisfiability models look like for formulas near the phase transition. In Section 3.3.3, we have already shown that satisfiable *QHORN* formulas have K_2 models. Furthermore, we have made the observation that existential variables with a satisfying assignment of 0 when all universals are 1 can remain 0 for any other combination of values that the universals may have. With our experiments, we want to confirm that even near the phase transition, random *QHORN* formulas have satisfiability models with a large fraction of constant functions $f_{y_i} = 0$. We also want to find out whether the remaining non-zero functions all tend to be quite short and simple as well, or whether we can also expect in each model a few more complicated functions.

At this point, it is useful to introduce a measurement for the complexity of individual satisfiability model functions and models as a whole. To enable future work beyond Horn formulas, this measurement should not be specific to K_2 functions. According to the definition of satisfiability models (Def. 2.7.1), each function f_{y_i} in a model can depend on all universal variables whose quantifiers precede the quantifier of y_i. In many typical instances of random or structured *QBF* formulas, however, the value of y_i is influenced by just a small subset of those universals. This is an important observation which will be fundamental to the discussion in Chapters 4 and 5 as well. One simple approach to measure the complexity of a model function f_{y_i} is therefore to determine the number of variables on which it actually depends. We call this number α the *actual arity* of f_{y_i}. In the case of *QHORN* formulas with K_2 satisfiability models, this is also an appropriate measure for the length of f_{y_i}, because in K_2 functions, variables occur at most once. For example, let $\Phi = \forall x_1, x_2, x_3 \exists y\ \phi$ with the satisfiability model $M_\Phi = (f_y)$ and $f_y(x_1, x_2, x_3) = x_1 \wedge x_3$. Then the actual arity of f_y is $\alpha = 2$, and that is also the formula length of f_y.

We now obtain a *satisfiability model distribution* for a formula Φ by counting for each possible arity value the number of functions in the satisfiability model M_Φ with this actual arity. The cases $f_{y_i} = 0$ and $f_{y_i} = 1$ are recorded separately as $\alpha = F$ and $\alpha = T$. The concept can easily be extended to sets of formulas $\Phi_1, ..., \Phi_k$ with satisfiability models $M_{\Phi_1}, ..., M_{\Phi_k}$ by counting functions among all those models.

For example, let Φ_1, Φ_2 be two formulas with the following satisfiability models:

$$\begin{aligned} M_{\Phi_1} &= (f_{y_1}(x_1,x_2) = x_2,\ f_{y_2}(x_1,x_2) = 0) \\ M_{\Phi_2} &= (f_{y_1}(x_1) = x_1,\ f_{y_2}(x_1,x_2,x_3) = x_1 \wedge x_2) \end{aligned}$$

Then we have the model distribution

$$\delta = \{(\alpha = F,\ 1),\ (\alpha = T,\ 0),\ (\alpha = 1,\ 2),\ (\alpha = 2,\ 1),\ (\alpha = 3,\ 0),\ ...\}$$

which should be read as: 1 constant function $f = 0$, no constant function $f = 1$, 2 functions with arity 1, 1 function with arity 2, etc.

Please notice that satisfiability models are not unique, and a single formula may have several different models. In fact, it is easy to show that *QHORN* formulas have the property that the intersection of two satisfiability models is again a satisfiability model [KBSZ07]. We have addressed this ambiguity by computing the models with the algorithm from [KBSZ04] rather than choosing a faster, but also less predictive algorithm like the one presented in Section 3.7.2 which assembles the models from arbitrary solutions to propositional subproblems and has therefore an inherent degree of freedom. Not only does the KBSZ algorithm produce completely predetermined models without any freedom of choice, but the computed models are also minimal in the sense that model functions are preferably set to constant 0 if possible. The algorithm determines for all existentials y_i the set of $\exists$-unit clauses with positive y_i which are derivable through Q-pos-unit resolution. A model function f_{y_i} is then composed from those universal variables which occur in all the derivable positive $\exists$-unit clauses with y_i. If no such unit clauses are derivable, we let $f_{y_i} = 0$. And if the derivable units have distinct universals, we assign $f_{y_i} = 1$.

In our experiments, we have used the FCL2-U generation model to produce lots of random *QHORN* formulas with identical parameters. We have extracted the satisfiable formulas and computed their satisfiability models and the resulting distribution. Figure 3.5 shows the model distribution we obtain for 200 formulas Φ with $\forall^*\exists^*$ prefix and $n = 500$ variables per quantifier block, i.e. Φ is of the form $\Phi = \forall x_1 ... \forall x_{500} \exists y_1 ... \exists y_{500}\ \phi$. The clause size was $h = 5$, and we chose $\frac{q}{n} = 5$, which is near the phase transition.

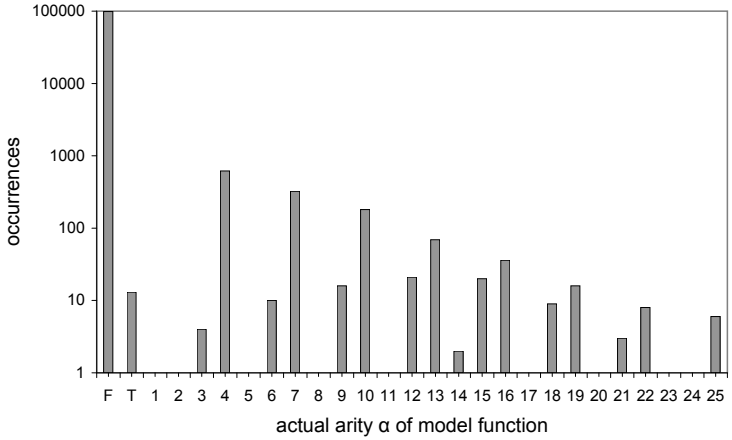

Figure 3.5.: Distribution of model functions for 200 random $\forall\exists$*HORN* formulas with $n = 500$ variables per quantifier block, clause size $h = 5$ and ratio $\frac{q}{n} = 5$

With 200 satisfiable formulas and 500 existentials each, the diagram includes a total of 100,000 satisfiability model functions. We can observe that the majority of them (98638) is simply the Boolean constant 0, which confirms our initial assumption on the basis of the model structure. On average, there are only 6.8 out of 500 existential variables per formula with a non-zero model function, which means they occur in derivable positive $\exists$-unit clauses. On the other hand, some of those functions have rather large arities up to 25. Thus we have an interesting imbalance where most of the model functions are constants, whereas some single existential variables per formula depend on many universals. Accordingly, our formulas were quite challenging for state-of-the-art *QBF* solvers which did not recognize Horn formulas.

There is an interesting pattern about the arity of the non-zero model functions: we can recognize that there are peaks at 4, 7, 10, 13, etc. with period 3. In general, for clause length h, the peaks are at $h-1+i(h-2)$ for $i \in \mathbb{N}_0$. To explain this, we must examine how positive $\exists$-units are produced by Q-pos-unit resolution. Our investigation is by induction on the length of Q-pos-unit resolution derivations. For the induction base, notice that the formula already has

some initial positive $\exists$-unit clauses with $h-1$ universals each. Now consider a clause $C = (y \vee \neg y_{j_1} \vee \ldots \vee \neg y_{j_m} \vee \neg x_{j_{m+1}} \vee \ldots \vee \neg x_{j_{h-1}})$ with one positive existential, $0 \leq m \leq h-1$ negative existentials and $h-m-1$ universals. We eliminate each existential literal $\neg y_{j_i}$ by resolving on it with a positive $\exists$-unit clause over y_{j_i} and n_i universals. If all universals are distinct, the resulting clause C' has $n = \sum_{i=1}^{m} n_i + h - m - 1$ universals. By induction, we have $n_i = h - 1 + c_i(h-2)$ for some $c_i \in \mathbb{N}_0$. Then $n = m(h-1) + c(h-2) + h - m - 1$ for an integer $c \geq 0$. We can rewrite this as $n = m(h-2) + m + c(h-2) + h - m - 1 = (m+c)(h-2) + h - 1$. Clearly, $n = h - 1 + \tilde{c}(h-2)$ for some integer $\tilde{c} \in \mathbb{N}_0$.

Notice that the argument in the previous paragraph assumes that all universals are distinct when resolving clauses. We can certainly expect this to be the case when n is large in comparison to the length of the clauses to be resolved. After a few resolution steps, however, clauses are getting increasingly large, and it becomes more likely that two resolvents have some universal variables in common. This explains why in Figure 3.5, the likelihood for arities in between the periodic peaks increases for larger arities, i.e. for longer Q-pos-unit resolution derivations. For example, we have more functions with arity 12 than with arity 9. Moving even further to the right, beyond arity 16, the number of occurrences is finally decreasing for all arities, because longer resolution derivations are less probable.

We can conclude that random *QHORN* formulas with fixed clause length show a typical sat-unsat-transition behavior, but even near the phase transition, they have fairly simple satisfiability models with characteristic patterns. It should be emphasized that this result is specific to random formulas, where complex Q-pos-unit resolution derivations are unlikely in satisfiable instances as they quickly lead to unsatisfiability. On the other hand, structured Horn formulas can be expected to have longer chains of Q-pos-unit resolution, starting with a considerable number of initial $\exists$-units (the *facts*). The structure of the problem will make sure that the initial units are not already unsatisfiable, which avoids the problems of the birthday paradox kind that we encountered for random formulas. Another modification which makes models significantly more complex will be discussed in the next section: quantified Horn formulas with free variables.

3.5. Equivalence Models for *QHORN** Formulas

As mentioned earlier, quantified Boolean formulas can be used to represent propositional formulas in a potentially shorter form. Equivalence is preserved by using free variables which correspond to the propositional variables in the original formula. Free variables are therefore a powerful feature of quantified Boolean formulas, but some of the concepts established for closed *QBF* formulas must be adapted. In particular, satisfiability models are only defined for formulas without free variables. The restriction can often be circumvented by considering fixed assignments to the free variables and then treating the formula with fixed free variables as a closed formula. This trick will later allow us to establish many of our results also for formulas with free variables, in particular the elimination of universal quantifiers (Section 3.6.2) and our new satisfiability testing algorithm (Section 3.7.1). But for other problems, such as investigations of expressiveness, working with fixed assignments to the free variables is not sufficient.

The goal of this section is to generalize our results on satisfiability models to "native" *QHORN** equivalence models. According to their definition in Section 2.7, equivalence models consist of functions which do not only depend on dominating universal variables, but also on all the free variables. In addition, equivalence models must lead to a tautology for each satisfying assignment to the free variables, which in general makes them more complex than satisfiability models.

3.5.1. Beyond K_2 Functions

We have seen that closed quantified Horn formulas have K_2 satisfiability models where the model functions are conjunctions of positive universal variables. It is easy to show that K_2 is not sufficient for *QHORN** equivalence models, even if we restrict ourselves to existentially quantified formulas as in the following proof.

Lemma 3.5.1. *There exist QHORN* formulas which do not have K_2 equivalence models.*

Proof:
Consider $\Phi \in QHORN^*$ with

$$\Phi(a_1,a_2,b_1,b_2) = \exists y\,(a_1 \vee \neg a_2 \vee \neg y) \wedge (y \vee \neg b_1) \wedge (y \vee \neg b_2)\ .$$

Expanding the existentially quantified variable y reveals:

$$\Phi \approx \Psi := (a_1 \vee \neg a_2 \vee \neg b_1) \wedge (a_1 \vee \neg a_2 \vee \neg b_2)$$

We can see that $M = (f_y)$ with $f_y(a_1,a_2,b_1,b_2) = a_1 \vee \neg a_2$ or $f_y(a_1,a_2,b_1,b_2) = b_1 \vee b_2$ are possible equivalence models. But there is no such $f_y \in K_2$: we can verify that $f_y = 0$, $f_y = 1$ or $f_y = b_1$ are no equivalence models. In all other cases, substituting $f_y = \bigwedge_{i\in I} z_i$ for y in the clause $(y \vee \neg b_1)$ will produce a binary clause $z_j \vee \neg b_1$ which is not in Ψ. □

While the formula in the proof does not have a K_2 equivalence model, the model function can be written as disjunction of positive variables. But we can easily show with an analogous proof that disjunction alone is not sufficient either. The question then is how we can generalize K_2 satisfiability models to equivalence models for $QHORN^*$ formulas. We have managed to come up with the following answer: the model functions are now conjunctions *and* disjunctions of *positive* universals and free variables. Thus it seems that the absence of negation in the model functions is a characteristic feature of quantified Horn formulas which is still preserved when free variables are allowed.

3.5.2. Monotone Models

More formally, we have been able to prove that quantified Horn formulas have monotone equivalence models. We start by defining what monotony means in this context.

Definition 3.5.2. *(Monotone Boolean Function)*
Let $\mathbf{x} = (x_1,...,x_n)$, $\mathbf{x}' = (x'_1,..,x'_n) \in \{0,1\}^n$, *and let* $f : \{0,1\}^n \to \{0,1\}$ *be a Boolean function. Then* f *is* ***monotone*** *if and only if* $\mathbf{x} \leq \mathbf{x}'$ *implies* $f(\mathbf{x}) \leq f(\mathbf{x}')$, *with the canonical ordering* $0 \leq 1$ *and* $\mathbf{x} \leq \mathbf{x}'$ *whenever* $x_i \leq x'_i$ *for all i.*

We usually represent the Boolean functions from which equivalence models are composed as propositional formulas. This leads to the following equivalent characterization of monotony (based on [Weg87]).

Proposition 3.5.3. *A Boolean function $f : \{0,1\}^n \to \{0,1\}$ is monotone if and only if it can be represented as a propositional formula F which contains only positive literals and the reduced operator set $\{\wedge, \vee\}$. We also allow $F = 0$ resp. $F = 1$.*

In the remainder of this section, we will always use this characterization of monotony.

Definition 3.5.4. *(Monotone Equivalence Model)*
*Let $M = (f_{y_1}, ..., f_{y_m})$ be an equivalence model for a quantified Boolean formula $\Phi \in QBF^*$. Then M is a **monotone equivalence model** if and only if the functions f_{y_i}, $1 \leq i \leq m$, are monotone.*

Notice that when we substitute an arbitrary monotone model M for the existential variables, the formula $\Phi[\mathbf{y}/M]$ may not be in *CNF* anymore. Of course, it can be transformed into *CNF* with the laws of associativity and distributivity and De Morgan's laws, but another problem may then occur: the resulting *CNF* formula is not necessarily a Horn formula.

In our proof, however, the construction of the model assures that $\Phi[\mathbf{y}/M]$ is a quantified Horn formula when transformed into *CNF*. The class of non-*CNF* formulas that may be transformed into *CNF* formulas with the Horn property shall be denoted with $QHORN^*_L$ as defined below.

Definition 3.5.5. *($QHORN^*_L$)*
With $QHORN^*_L$, we denote the class of quantified Boolean formulas $\Phi \in QBF^*$ for which there exist $\Phi' \in QHORN^*$ such that Φ' can be obtained from Φ by applying the laws of associativity and distributivity and De Morgan's laws.

3.5.3. $QHORN^*$ Equivalence Models Are Monotone

We now show that a quantified Horn formula always has a monotone equivalence model. In the proof, we inductively construct such a model for any $\Phi \in QHORN^*$.

Theorem 3.5.6. *Any formula $\Phi \in QHORN^*$ has a monotone equivalence model $M = (f_{y_1}, ..., f_{y_m})$. Moreover, M can be chosen such that $\Phi[\mathbf{y}/M] \in QHORN^*_L$.*

Proof:
If $\Phi(\mathbf{z})$ is unsatisfiable, there is a $\{0,1\}$-equivalence model, and therefore a monotone equivalence model M with $\Phi[\mathbf{y}/M] \in QHORN_L^*$. For the remainder of this proof, we assume the satisfiability of the input formula and prove the theorem by induction on the number of quantifiers.

For $k = 1$, we have a formula with one quantifier, which may be universal or existential. If $\Phi(\mathbf{z}) = \forall x_1\, \phi(x_1, \mathbf{z})$ with a propositional formula ϕ, then the empty model $M = ()$ is a monotone equivalence model for Φ.

The second case in which the quantifier is existential is more interesting. Suppose Φ is given as $\Phi(\mathbf{z}) = \exists y_1\, \phi(y_1, \mathbf{z})$ with a propositional formula ϕ. If y_1 or $\neg y_1$ occurs in $\phi(y_1, \mathbf{z})$ as a unit clause, define $f_{y_1} = 1$ or $f_{y_1} = 0$, respectively.
If y_1 occurs only positively or only negatively in $\phi(y_1, \mathbf{z})$, let $f_{y_1} = 1$ or $f_{y_1} = 0$.
If y_1 occurs both positively and negatively, let $\neg a_{i,1} \vee \ldots \vee \neg a_{i,s_i} \vee y_1 =: \neg A_i \vee y_1$ be the clauses in which y_1 occurs positively ($1 \leq i \leq c_{pos}$, where c_{pos} is the number of those clauses). Analogously, let $b_{j,1} \vee \ldots \vee b_{j,t_j} \vee \neg y_1 =: B_j \vee \neg y_1$ be the clauses in which y_1 occurs negatively ($1 \leq j \leq c_{neg}$). Finally, let C be the clauses which contain neither y_1 nor $\neg y_1$. Clauses which contain both y_1 and $\neg y_1$ are tautological and can therefore be removed from the formula. If y_1 only occurs in tautological clauses, we can also remove that variable itself.
We now define the model of y_1. The idea is to choose a model such that tautological clauses are created when f_{y_1} is substituted for positive instances of y_1, while substituting f_{y_1} for the negative instances of y_1 produces the expansion $\phi(0, \mathbf{z}) \vee \phi(1, \mathbf{z})$ of the existentially quantified formula $\exists y_1\, \phi(y_1, \mathbf{z})$. That can be accomplished with the following definition:

$$f_{y_1} = \bigvee_{1 \leq i \leq c_{pos}} A_i = \bigvee_{1 \leq i \leq c_{pos}} (a_{i,1} \wedge \ldots \wedge a_{i,s_i})$$

For a clause $\neg A_i \vee y_1$ in which y_1 occurs positively, we then obtain $\neg A_i \vee f_{y_1} = \neg A_i \vee A_1 \vee \ldots \vee A_i \vee \ldots \vee A_{c_{pos}}$, which contains both A_i and $\neg A_i$ and is thus tautological.
On the other hand, consider the set of clauses in which y_1 occurs negatively:

$$\begin{aligned} \left(\textstyle\bigwedge_j (B_j \vee \neg y_1)\right)[y_1/f_{y_1}] &= \textstyle\bigwedge_j (B_j \vee \neg (\bigvee_i A_i)) \\ &\approx \textstyle\bigwedge_j (B_j \vee (\bigwedge_i \neg A_i)) \\ &\approx \textstyle\bigwedge_{i,j} (\neg A_i \vee B_j) \end{aligned}$$

The clauses C which do not contain y_1 (respectively $\neg y_1$) remain unchanged. As motivated before, the resulting formula $\Phi[y_1/f_{y_1}] \approx \bigwedge_{i,j}(\neg A_i \vee B_j) \wedge C$ is the expansion of the existentially quantified formula $\exists y_1\, \phi(y_1, \mathbf{z})$, which can be seen as follows:

$$\begin{aligned}
\exists y_1\, \phi(y_1, \mathbf{z}) &\approx \phi(0, \mathbf{z}) \vee \phi(1, \mathbf{z}) \\
&\approx (\textstyle\bigwedge_i \neg A_i \wedge C) \vee \left(\bigwedge_j B_j \wedge C\right) \\
&\approx \left((\textstyle\bigwedge_i \neg A_i) \vee \left(\bigwedge_j B_j\right)\right) \wedge C \\
&\approx \textstyle\bigwedge_{i,j}(\neg A_i \vee B_j) \wedge C
\end{aligned}$$

This proves that $M = (f_{y_1})$ is an equivalence model for $\Phi(\mathbf{z})$. Please notice that $\bigwedge_{i,j}(\neg A_i \vee B_j) \wedge C$ is a Horn formula, because the $\neg A_i$ contain only negative literals, and each B_j has at most one positive literal. Thus, $\Phi[y_1/f_{y_1}] \in QHORN^*_L$.

Now let $k > 1$. Again, we have two cases: the outer quantifier may either be universal or existential. If it is universal, Φ has the form $\Phi(\mathbf{z}) = \forall x_k \Phi'(x_k, \mathbf{z})$, where Φ' is a formula with $k-1$ quantifiers. If $\Phi \in QHORN^*$, then also $\Phi' \in QHORN^*$, and by the induction hypothesis, Φ' has a monotone equivalence model $M_{\Phi'}$ with $\Phi'[\mathbf{y}/M_{\Phi'}] \in QHORN^*_L$. $M_{\Phi'}$ is also a monotone equivalence model for Φ, because $\Phi' \approx \Phi'[\mathbf{y}/M_{\Phi'}]$ implies:

$$\begin{aligned}
\Phi(\mathbf{z}) \;=\; \forall x_k \Phi'(x_k, \mathbf{z}) \;\approx\; \forall x_k (\Phi'(x_k, \mathbf{z})[\mathbf{y}/M_{\Phi'}]) \;&=\; (\forall x_k \Phi'(x_k, \mathbf{z}))[\mathbf{y}/M_{\Phi'}] \\
&= \Phi(\mathbf{z})[\mathbf{y}/M_{\Phi'}]
\end{aligned}$$

Obviously, $\Phi[\mathbf{y}/M_{\Phi'}] \in QHORN^*_L$ as well.

In the second case, the outer quantifier is existential, and Φ has the form $\Phi(\mathbf{z}) = \exists y_k \Phi'(y_k, \mathbf{z})$. Notice that y_k is a free variable in Φ'. If Φ' contains only universal quantifiers, we can remove all of them, as they do not dominate any existentially quantified variables. We are then left with only one existential variable and can proceed as in the induction base. For the remainder of this proof, we assume that Φ' contains at least one existentially quantified variable.
As above, Φ' is a formula with $k-1$ quantifiers, and according to the induction hypothesis, it has a monotone equivalence model $M_{\Phi'} = (f'_{y_1}, ..., f'_{y_{k-1}})$ with $\Phi'[\mathbf{y}/M_{\Phi'}] \in QHORN^*_L$. $\Phi'(y_k, \mathbf{z}) \approx \Phi'(y_k, \mathbf{z})[\mathbf{y}/M_{\Phi'}]$ implies

$$\Phi(\mathbf{z}) = \exists y_k \Phi'(y_k, \mathbf{z}) \approx \exists y_k (\Phi'(y_k, \mathbf{z})[\mathbf{y}/M_{\Phi'}])\ .$$

Now, $\Phi'[\mathbf{y}/M_{\Phi'}] \in QHORN^*_L$ means that there exists $\Phi''(\mathbf{z}) \in QHORN^*$ such that $\Phi''(\mathbf{z}) \approx \exists y_k (\Phi'(y_k, \mathbf{z})[\mathbf{y}/M_{\Phi'}])$. Under the assumption that Φ' contains at

least one existential variable, $\Phi'[\mathbf{y}/M_{\Phi'}]$ has less than $k-1$ quantifiers. Thus, Φ'' has less than k quantifiers, and only the outermost is existential. By the induction hypothesis, it has a monotone equivalence model $M_{\Phi''} = (f''_{y_k})$ with $\Phi''[y_k/f''_{y_k}] \in QHORN^*_L$.
We now combine $M_{\Phi'} = (f'_{y_1}, ..., f'_{y_{k-1}})$ and $M_{\Phi''} = (f''_{y_k})$ into a monotone equivalence model $M = (f_{y_1}, ..., f_{y_k})$ for the original formula Φ by assigning $f_{y_i} = f'_{y_i}[y_k/f''_{y_k}]$ for $1 \leq i \leq k-1$ and $f_{y_k} = f''_{y_k}$. It is obvious that M is monotone. Informally, it is also clear that M is an equivalence model for Φ, but the formal proof is somewhat tedious:

$$\begin{aligned}
\Phi(\mathbf{z}) &\approx \Phi''(\mathbf{z}) \\
&\approx \Phi''(\mathbf{z})[y_k/f''_{y_k}] \\
&\approx (\exists y_k\,(\Phi'(y_k,\mathbf{z})[y_1/f'_{y_1}, ..., y_{k-1}/f'_{y_{k-1}}]))[y_k/f''_{y_k}] \\
&= (\exists y_k\,\Phi'(y_k,\mathbf{z}))[y_1/f'_{y_1}[y_k/f''_{y_k}], ..., y_{k-1}/f_{y_{k-1}}[y_k/f''_{y_k}], y_k/f''_{y_k}] \\
&= (\exists y_k\,\Phi'(y_k,\mathbf{z}))[y_1/f_{y_1}, ..., y_k/f_{y_k}] \\
&= \Phi(\mathbf{z})[\mathbf{y}/M]
\end{aligned}$$

$\Phi(\mathbf{z})[\mathbf{y}/M] \in QHORN^*_L$, because $\Phi''(\mathbf{z})[y_k/f''_{y_k}] \in QHORN^*_L$ and $\Phi(\mathbf{z})[\mathbf{y}/M] \approx \Phi''(\mathbf{z})[y_k/f''_{y_k}]$. □

The previous result reveals the structure of equivalence models for *QHORN** formulas. Unfortunately, the proof itself does not lead to a feasible algorithm for finding those equivalence models. The problem with the algorithm suggested by the proof is that the formula which is being worked on may blow up exponentially. As the algorithm moves step by step from the innermost quantifiers to the outermost quantifiers, the model found in the previous step is always substituted into the given formula which is then re-transformed into *CNF*. In the end, the remaining universal quantifiers can easily be dropped, and we obtain a propositional *CNF* formula, which must be exponentially longer for certain formulas due to the previously mentioned result that $CNF <_{poly-length} QHORN^*$. We will see in Section 3.8 that there is a close connection between quantified Horn formulas and Boolean circuits in which inner nodes are allowed to have more than one outgoing edge. It is widely assumed that these are exponentially more concise than arbitrary propositional formulas. Then there would also be *QHORN** formulas for which every equivalent propositional formula has exponential size, which in turn implies that such formulas cannot have polynomial-size equivalence models.

3.6. Elimination of Universal Quantifiers

In the previous section, we have just encountered once again the infeasibility of eliminating the existential quantifiers in a *QHORN** formula, so we now turn our attention to the universal quantifiers. Section 3.3 has revealed that cases where at most one of the universal variables is false completely determine the behavior of the existential quantifiers, which represents a very strong restriction on the expressive power of universal quantifiers in Horn formulas. So far, we have proved this for closed formulas, but we now extend it to *QHORN** by considering fixed assignments to the free variables. We show that this result significantly facilitates eliminating the universal quantifiers, which allows us to remove all of them and still avoid the exponential growth that is associated with eliminating both kinds of quantifiers. Let us begin with the following definition:

Definition 3.6.1. *(∃HORN*, ∃k-HORN*, ∃BF*)*
A formula $\Phi \in$ QHORN* (Qk-HORN* for a constant $k \geq 2$, respectively) is an ***existentially quantified (k-)Horn formula*** *with free variables if it is of the form $\Phi(\mathbf{z}) = \exists y_1 ... \exists y_m \, \phi(\mathbf{z}, y_1, ..., y_m)$, $m \geq 0$, i.e. if it does not contain universally quantified variables. The class of all such formulas we denote by ∃HORN* (∃k-HORN*, respectively). Analogously, we let ∃BF* denote purely* ***existentially quantified Boolean formulas****.*

The goal of our investigation is to transform an arbitrary formula in *QHORN** into an equivalent formula in ∃*HORN** with a polynomial increase in length. Our idea is to perform universal expansion on all universal quantifiers at once. We first consider this transformation in general for *QBF** formulas, in which case it will clearly show exponential blowup, and then prove that it can be simplified to polynomial size for *QHORN**.

3.6.1. Basic Universal Expansion Algorithm for *QBF**

As suggested in the introductory chapter, expanding a single universal quantifier $\forall x$ is rather straightforward when we consider the quantifier as an abbreviation for the universally quantified variable x being true in one case and being false in the other case. That means we must generate two copies of the original matrix and can then drop x from the prefix:

$$Q\mathbf{v} \forall x \; \phi(x, \mathbf{v}, \mathbf{z}) \approx Q\mathbf{v} \; \phi(0, \mathbf{v}, \mathbf{z}) \wedge \phi(1, \mathbf{v}, \mathbf{z})$$

Special care must be taken if there are existential quantifiers that are in the scope of $\forall x$. In the formula $Q\mathbf{v}\forall x\exists y_1...\exists y_m\ \phi(x,y_1,...,y_m,\mathbf{v},\mathbf{z})$, each of the innermost existentials can be assigned at least two different truth values: one for the case $x=0$ and one for $x=1$. As $(\exists y\ \Phi(0,y))\wedge(\exists y\ \Phi(1,y))\not\approx\exists y\ (\Phi(0,y)\wedge\Phi(1,y))$, such existential variables y_i need to be duplicated as well to reflect the degree of freedom to have different values of y_i for different values of the preceding x. For example, in the formula $\exists y_1\forall x\exists y_2\,\phi(x,y_1,y_2)$, the choice for the existential variable y_2 depends on the value of x. We must therefore introduce two separate instances $y_2^{(0)}$ and $y_2^{(1)}$ of the original variable y_2, where $y_2^{(0)}$ is used in the copy of the matrix for $x=0$, and analogously $y_2^{(1)}$ for $x=1$. We obtain the result $\exists y_1\exists y_2^{(0)}\exists y_2^{(1)}\,\phi(0,y_1,y_2^{(0)})\wedge\phi(1,y_1,y_2^{(1)})$. In general, the expansion is as follows:

$$\begin{aligned} Q\mathbf{v}\forall x\exists y_1...\exists y_m\ \phi(x,y_1,...,y_m,\mathbf{v},\mathbf{z}) \ \approx\ & Q\mathbf{v}\exists y_1^{(0)}...\exists y_m^{(0)}\exists y_1^{(1)}...\exists y_m^{(1)} \\ & \phi(0,y_1^{(0)},...,y_m^{(0)},\mathbf{v},\mathbf{z}) \\ \wedge\ & \phi(1,y_1^{(1)},...,y_m^{(1)},\mathbf{v},\mathbf{z}) \end{aligned}$$

At this point, we do not provide a formal proof of the above equivalence. On the one hand, it immediately follows from our initial remarks, and on the other hand, we will later prove an even more useful generalization of it in Theorem 5.3.1, Section 5.3.1.

Existing applications of universal expansion [AB02, Bie05] perform only such single expansion steps and alternate them with other techniques like Q-resolution or quantifier miniscoping to prevent the expanded formula from growing too quickly due to repeated duplications of the matrix. In contrast, we now consider the complete expansion of all universal quantifiers at once and investigate how to simplify it for *QHORN** formulas. In order to do so, we first need a detailed formalization of the general case.

Let $\Phi\in QBF^*$ with $\Phi(\mathbf{z})=\forall X_1\exists Y_1...\forall X_r\exists Y_r\,\phi(X_1,...,X_r,Y_1,...,Y_r,\mathbf{z})$ be the formula whose universal quantifiers we want to expand. $X_i=(x_{i,1},...,x_{i,n_i})$ and $Y_i=(y_{i,1},...,y_{i,m_i})$ ($n_i\geq 1$ and $m_i\geq 1$, $i=1,...,r$, $r\geq 1$) are the quantifier blocks in the prefix, and ϕ is the propositional matrix. Without loss of generality, we assume that the outermost quantifiers are universal. If they were existential, we could treat these existentially quantified variables as free variables, and the outermost quantifiers in the remaining prefix would then be universal. Furthermore, we assume that the innermost quantifiers are existential.

When expanding all universals in $\Phi(\mathbf{z})$, starting with the innermost, we obtain:

$$\begin{aligned}\Phi_{\exists\text{exp}}(\mathbf{z}) := \bigwedge_{A_1\in\{0,1\}^{n_1}} \Big(&\exists Y_1^{A_1} \\ &\bigwedge_{A_2\in\{0,1\}^{n_2}} \Big(\exists Y_2^{A_1A_2} \\ &\quad\vdots \\ &\bigwedge_{A_r\in\{0,1\}^{n_r}} \Big(\exists Y_r^{A_1\ldots A_r}\, \phi(A_1\ldots A_r, Y_1^{A_1}\ldots Y_r^{A_1\ldots A_r}, \mathbf{z})\Big)\ldots\Big)\Big)\end{aligned}$$

The tuples A_i represent the possible truth assignments to the universal variables $x_{i,1},\ldots,x_{i,n_i}$. The expression $\bigwedge_{A_i\in\{0,1\}^{n_i}}$ should be understood as a conjunction of 2^{n_i} clauses, one for each truth assignment. Finally, $\exists Y_i^{A_1\ldots A_i}$ is an abbreviation for $\exists y_{i,1}^{A_1\ldots A_i}\ldots\exists y_{i,m_i}^{A_1\ldots A_i}$, the copies of the i-th block of existential quantifiers. The additional index $A_1\ldots A_i$ is used to tag each copy with the values of the preceding universal variables. Its purpose is to have a unique name for each of those copies. For example, four copies of $y_{i,j}$ would be named $y_{i,j}^{(0,0)}$, $y_{i,j}^{(0,1)}$, $y_{i,j}^{(1,0)}$ and $y_{i,j}^{(1,1)}$.

Using induction on the number of blocks of universal quantifiers, it is possible to show that $\Phi(\mathbf{z}) \approx \Phi_{\exists\text{exp}}(\mathbf{z})$. We omit this, as it is quite obvious that $\Phi_{\exists\text{exp}}$ is simply the formalization of the elimination algorithm described above.

Here is an example: the formula

$$\Phi(\mathbf{z}) = \forall x_1\exists y_1\forall x_2\forall x_3\exists y_2\, \phi(x_1,x_2,x_3,y_1,y_2,\mathbf{z})$$

is expanded to

$$\begin{aligned}\Phi_{\exists\text{exp}}(\mathbf{z}) = \exists y_1^{(0)}(&\exists y_2^{(0,0,0)}\phi(0,0,0,y_1^{(0)},y_2^{(0,0,0)},\mathbf{z}) \wedge \exists y_2^{(0,0,1)}\phi(0,0,1,y_1^{(0)},y_2^{(0,0,1)},\mathbf{z}) \wedge \\ &\exists y_2^{(0,1,0)}\phi(0,1,0,y_1^{(0)},y_2^{(0,1,0)},\mathbf{z}) \wedge \exists y_2^{(0,1,1)}\phi(0,1,1,y_1^{(0)},y_2^{(0,1,1)},\mathbf{z}))\wedge \\ \exists y_1^{(1)}(&\exists y_2^{(1,0,0)}\phi(1,0,0,y_1^{(1)},y_2^{(1,0,0)},\mathbf{z}) \wedge \exists y_2^{(1,0,1)}\phi(1,0,1,y_1^{(1)},y_2^{(1,0,1)},\mathbf{z}) \wedge \\ &\exists y_2^{(1,1,0)}\phi(1,1,0,y_1^{(1)},y_2^{(1,1,0)},\mathbf{z}) \wedge \exists y_2^{(1,1,1)}\phi(1,1,1,y_1^{(1)},y_2^{(1,1,1)},\mathbf{z}))\end{aligned}$$

$\Phi_{\exists\text{exp}}$ is not in prenex form. This would be easy to fix by moving all quantifiers to the front. In the sample formula above, the prefix might then look like

$$\exists y_1^{(0)}\exists y_2^{(0,0,0)}\exists y_2^{(0,0,1)}\exists y_2^{(0,1,0)}\exists y_2^{(0,1,1)}\exists y_1^{(1)}\exists y_2^{(1,0,0)}\exists y_2^{(1,0,1)}\exists y_2^{(1,1,0)}\exists y_2^{(1,1,1)}\,.$$

For clarity's sake, we did not consider this in the general formula $\Phi_{\exists\text{exp}}$.

As the expansion example above demonstrates, the resulting formula is rather voluminous. In general, it is exponentially longer: if there are n universal quantifiers in an input formula Φ, its expansion $\Phi_{\exists\text{exp}}$ contains 2^n copies of the original matrix. Fortunately, we can discard most of these copies in the special case of quantified Horn formulas.

3.6.2. Universal Expansion for $QHORN^*$ Formulas

Definition 3.6.2. *(Universal Expansion for QHORN*)*
Let $\Phi \in QHORN^*$ with

$$\Phi(\mathbf{z}) = \forall X_1 \exists Y_1 ... \forall X_r \exists Y_r \, \phi(X_1, ..., X_r, Y_1, ..., Y_r, \mathbf{z})$$

where $X_i = (x_{i,1}, ..., x_{i,n_i})$ and $Y_i = (y_{i,1}, ..., y_{i,m_i})$ ($n_i \geq 1$ and $m_i \geq 1$, $i = 1, ..., r$, $r \geq 1$), be a quantified Horn formula whose outermost quantifiers are universal and whose innermost quantifiers are existential.

Then we define the formula $\Phi_{\exists\text{poly}}(\mathbf{z})$ as

$$\begin{aligned}\Phi_{\exists\text{poly}}(\mathbf{z}) := \bigwedge_{A_1 \in Assign_1} \Big(& \exists Y_1^{A_1} \\ & \bigwedge_{A_2 \in Assign_2(A_1)} \Big(\exists Y_2^{A_1 A_2} \\ & \qquad \vdots \\ & \bigwedge_{A_r \in Assign_r(A_1 ... A_{r-1})} \Big(\exists Y_r^{A_1 ... A_r} \, \phi(A_1 ... A_r, Y_1^{A_1} ... Y_r^{A_1 ... A_r}, \mathbf{z}) \Big) ... \Big) \Big)\end{aligned}$$

with the restricted set of possible assignments

$$Assign_1 = Z_{\leq 1}(n_1)$$

$$Assign_i(A_1, ..., A_{i-1}) = \begin{cases} Z_{\leq 1}(n_i) & \text{, if } A_1 ... A_{i-1} = \{1\}^{n_1 + ... + n_{i-1}} = (1, ..., 1) \\ \{1\}^{n_i} & \text{, else} \end{cases}$$

The only difference between the formula $\Phi_{\exists\text{poly}}$ and the expansion $\Phi_{\exists\text{exp}}$ for general $QCNF^*$ formulas which was presented in the previous section is that for quantified Horn formulas, not all possible truth assignments to the universally quantified variables have to be considered. For Horn formulas, we discard assignments where more than one universal variable is false.

For the formula

$$\Phi(\mathbf{z}) = \forall x_1 \exists y_1 \forall x_2 \forall x_3 \exists y_2 \, \phi(x_1, x_2, x_3, y_1, y_2, \mathbf{z})$$

from the example in the previous section, we have

$$\begin{aligned}
\Phi_{\exists\text{poly}}(\mathbf{z}) \;=\; \exists y_1^{(0)} \;\; & \exists y_2^{(0,1,1)} \phi(0,1,1,y_1^{(0)}, y_2^{(0,1,1)}, \mathbf{z}) \wedge \\
\exists y_1^{(1)} (\; & \exists y_2^{(1,0,1)} \phi(1,0,1,y_1^{(1)}, y_2^{(1,0,1)}, \mathbf{z}) \wedge \\
& \exists y_2^{(1,1,0)} \phi(1,1,0,y_1^{(1)}, y_2^{(1,1,0)}, \mathbf{z}) \wedge \\
& \exists y_2^{(1,1,1)} \phi(1,1,1,y_1^{(1)}, y_2^{(1,1,1)}, \mathbf{z}))
\end{aligned}$$

Before we can prove that $\Phi_{\exists\text{poly}}$ is indeed equivalent to Φ when $\Phi \in \mathit{QHORN}^*$, we make a fundamental observation: for the special case of closed formulas $\Phi \in \mathit{QHORN}$, the satisfiability of $\Phi_{\exists\text{poly}}$ implies the existence of a $Z_{\leq 1}$-partial satisfiability model for Φ.

Lemma 3.6.3. *Let $\Phi \in \mathit{QHORN}$ be a quantified Horn formula without free variables, and let $\Phi_{\exists\text{poly}}$ be defined as above. If $\Phi_{\exists\text{poly}}$ is satisfiable then Φ has a $Z_{\leq 1}$-partial satisfiability model.*

Proof:
Let t be a satisfying truth assignment to the existentials in $\Phi_{\exists\text{poly}}$. This assignment t provides us with all the information needed to construct a $Z_{\leq 1}$-partial satisfiability model for Φ.
The idea is to assemble the truth assignments to the individual copies $y_{i,j}^{(x_{1,1},\ldots,x_{i,n_i})}$ of an existential variable $y_{i,j}$ into a common model function. It works as follows: let $y_{i,j}$ be an existential variable in Φ whose corresponding quantifier is preceded by the universal quantifiers $\forall x_{1,1} \ldots \forall x_{i,n_i}$. Then we define:

$$\begin{aligned}
f_{y_{i,j}}(x_{1,1}, \ldots, x_{i,n_i}) \;=\; & (\neg x_{1,1} \wedge x_{1,2} \wedge \ldots \wedge x_{i,n_i} \;\rightarrow t(y_{i,j}^{(0,1,\ldots,1)})) \\
\wedge \; & (x_{1,1} \wedge \neg x_{1,2} \wedge x_{1,3} \wedge \ldots \wedge x_{i,n_i} \;\rightarrow t(y_{i,j}^{(1,0,1,\ldots,1)})) \\
\wedge \; & \ldots \\
\wedge \; & (x_{1,1} \wedge \ldots \wedge x_{i,n_i-1} \wedge \neg x_{i,n_i} \;\rightarrow t(y_{i,j}^{(1,\ldots,1,0)})) \\
\wedge \; & (x_{1,1} \wedge \ldots \wedge x_{i,n_i} \;\rightarrow t(y_{i,j}^{(1,\ldots,1)}))
\end{aligned}$$

Now, the $f_{y_{i,j}}$ form a $Z_{\leq 1}$-partial satisfiability model for Φ, because for all $\mathbf{x} = (x_{1,1}, ..., x_{r,n_r})$ with $\mathbf{x} \in Z_{\leq 1}$, we have $f_{y_{i,j}}(x_{1,1}, ..., x_{i,n_i}) = t(y_{i,j}^{(x_{1,1},...,x_{i,n_i})})$ and $\phi(\mathbf{x}, t(y_{1,1}^{(x_{1,1},...,x_{1,n_1})}), ..., t(y_{r,m_r}^{(x_{1,1},...,x_{r,n_r})})) = 1$ as $\Phi_{\exists\text{poly}}$ is satisfiable. □

Using Lemma 3.6.3 in combination with Theorem 3.3.5, it is now easy to show that $\Phi_{\exists\text{poly}}$ is equivalent to Φ.

Theorem 3.6.4. *Let $\Phi \in QHORN^*$ be a quantified Horn formula with free variables, and let $\Phi_{\exists\text{poly}}$ be its poly-length universal expansion as in Definition 3.6.2. Then $\Phi_{\exists\text{poly}} \approx \Phi$.*

Proof:
The implication $\Phi(\mathbf{z}) \models \Phi_{\exists\text{poly}}(\mathbf{z})$ is obvious, as the clauses in $\Phi_{\exists\text{poly}}$ are just a subset of the clauses in $\Phi_{\exists\text{exp}}$, which in turn is equivalent to Φ.

The implication $\Phi_{\exists\text{poly}}(\mathbf{z}) \models \Phi(\mathbf{z})$ is more interesting. Assume that $\Phi_{\exists\text{poly}}(\mathbf{z}^*)$ is satisfiable for some fixed $\mathbf{z}^*$. With the free variables fixed, we can treat both $\Phi_{\exists\text{poly}}(\mathbf{z}^*)$ and $\Phi(\mathbf{z}^*)$ as closed formulas and apply Lemma 3.6.3 and the results from Section 3.3.1 as follows:
According to Lemma 3.6.3, the satisfiability of $\Phi_{\exists\text{poly}}(\mathbf{z}^*)$ implies that $\Phi(\mathbf{z}^*)$ has a $Z_{\leq 1}$-partial satisfiability model. On this partial model, we can apply the total completion from Definition 3.3.3 and Theorem 3.3.5 to obtain a (total) satisfiability model. The fact that $\Phi(\mathbf{z}^*)$ has a satisfiability model implies that $\Phi(\mathbf{z}^*)$ is satisfiable. □

In the definition of $\Phi_{\exists\text{poly}}$, we can observe that there is one instantiation of the matrix of the original formula for each possible assignment to the universal variables in which either all of those variables are true or exactly one of them is false. There are $n+1$ such assignments. Furthermore, the previous theorem has shown that $\Phi_{\exists\text{poly}}$ is equivalent to Φ, so we have the following corollary.

Corollary 3.6.5. *For any quantified Horn formula $\Phi \in QHORN^*$ with free variables, there exists an equivalent formula $\Phi' \in \exists HORN^*$ without universal quantifiers. The length of Φ' is bounded by $|\forall| \cdot |\Phi|$, where $|\forall|$ is the number of universal quantifiers in Φ, and $|\Phi|$ is the length of Φ.*

3.6.3. The $\exists HORN^*$ Transformation Algorithm

It is easy to see that our $\exists HORN^*$ transformation can be computed in polynomial time. Let $\Phi \in QHORN^*$ with $\Phi(\mathbf{z}) = \forall X_1 \exists Y_1 ... \forall X_r \exists Y_r \, \phi(X_1, ..., X_r, Y_1, ..., Y_r, \mathbf{z})$ where $X_i = (x_{i,1}, ..., x_{i,n_i})$ and $Y_i = (y_{i,1}, ..., y_{i,m_i})$ $(n_i, m_i \geq 1, i = 1, ..., r, r \geq 1)$, be a quantified Horn formula whose outermost quantifiers are universal and whose innermost quantifiers are existential. Listing 3.1 presents an algorithm to transform Φ into $\Phi_{\exists\text{poly}}$ as described above.

Listing 3.1: The $\exists HORN^*$ transformation algorithm

```
Input  Φ ∈ QHORN*,  Φ(z) = ∀X_1 ∃Y_1 ... ∀X_r ∃Y_r φ(X_1, ..., X_r, Y_1, ..., Y_r, z),
         where X_i = (x_{i,1}, ..., x_{i,n_i}) and Y_i = (y_{i,1}, ..., y_{i,m_i});

φ_∃poly = ∅;
for (i = 1 to r)
  for (j = 1 to n_i) A_{x_{i,j}} = 1;
for (i = 1 to r) {
  for (j = 1 to n_i) {
    A_{x_{i,j}} = 0;
    for (k = i to r)
      for (l = 1 to m_k) y'_{k,l} = new ∃–var;
    φ_∃poly = φ_∃poly ∪ φ[x/A_x, y/y']; // (*)
    A_{x_{i,j}} = 1;
  }
  for (l = 1 to m_i) y'_{i,l} = new ∃–var;
}
φ_∃poly = φ_∃poly ∪ φ[x/A_x, y/y']; // (*)
Φ_∃poly = ∃y' φ_∃poly;

Output  Φ_∃poly ∈ ∃HORN* with Φ_∃poly ≈ Φ.
```

In the main loop of the algorithm, one universal variable $x_{i,j}$ is given the value false, while all the others are true. For any such assignment $A_{\mathbf{x}}$, all existential variables which are dominated by $x_{i,j}$ (i.e. their corresponding quantifier follows $\forall x_{i,j}$) have to be replaced by independent new variables $\mathbf{y}'$. Then, the matrix of the original formula has to be duplicated, with $A_{\mathbf{x}}$ being substituted for $\mathbf{x}$ and $\mathbf{y}'$ being substituted for $\mathbf{y}$. After executing the main loop, one additional copy is needed for the case where all universal variables are true. Notice that we treat the existential variables as objects. If we let $y'_{i,j} = \texttt{new}\exists\texttt{-var}$ and use this variable

in multiple locations, then all share the same variable object, which means all those subformulas share that existential variable. The lines marked with (*) need time $O(|\Phi|)$. They are executed $n_1 + ... + n_r + 1 = |\forall| + 1$ times, so the algorithm in total requires time $O(|\forall| \cdot |\Phi|)$.

With this result, it is safe to say that the class $QHORN^*$ is not significantly more expressive than the class $\exists HORN^*$. Under the relation $=_{poly-time}$ from Definition 2.6.1, both classes are equivalent.

Corollary 3.6.6. $QHORN^* =_{poly-time} \exists HORN^*$.

3.7. Satisfiability Testing and Model Computation

The ability to eliminate all universal quantifiers from $QHORN^*$ formulas is not only a powerful tool for simplification, but also opens up the possibility to solve those formulas by transforming them into satisfiability-equivalent propositional formulas. Accordingly, we now present an efficient new algorithm for the $QHORN^*$ satisfiability problem and show how it can be extended to also compute the actual satisfiability models for true $QHORN$ formulas.

3.7.1. Solving $QHORN^*$ Formulas

Let $\Psi(\mathbf{z}) \in \exists HORN^*$ be an existentially quantified Horn formula of the form $\Psi(\mathbf{z}) = \exists y_1 ... \exists y_m \, \psi(y_1, ..., y_m, \mathbf{z})$. Then $\Psi(\mathbf{z})$ is satisfiable if and only if its matrix $\psi(y_1, ..., y_m, \mathbf{z})$ is satisfiable. The latter is a purely propositional formula, therefore a satisfiability solver for propositional Horn formulas can be used to determine the satisfiability of an arbitrary formula in $\exists HORN^*$. That makes $\exists HORN^*$ a suitable representation for satisfiability testing. And since we have just shown that we can efficiently transform arbitrary $QHORN^*$ formulas into $\exists HORN^*$, it appears attractive to always take this detour from $QHORN^*$ to $\exists HORN^*$. We then obtain the following algorithm for determining the satisfiability of a formula $\Phi \in QHORN^*$:

1. Transform Φ into $\Phi_{\exists \text{poly}} \in \exists HORN^*$ with $|\Phi_{\exists \text{poly}}| = O(|\forall| \cdot |\Phi|)$. This requires time $O(|\forall| \cdot |\Phi|)$ as discussed in Section 3.6.3.

2. Determine the satisfiability of $\phi_{\exists\text{poly}}$, which is the purely propositional matrix of $\Phi_{\exists\text{poly}}$. It is well known [DG84] that propositional Horn formulas can be solved in linear time, in this case $O(|\phi_{\exists\text{poly}}|) = O(|\forall| \cdot |\Phi|)$.

In total, the algorithm requires time $O(|\forall| \cdot |\Phi|)$. The best existing algorithm presented in [FKKB95, KBL99] has the same complexity, but that algorithm is significantly more complicated and cannot directly reuse existing propositional satisfiability solvers like this new algorithm does. The most important advantage, however, is that our algorithm is not only capable of deciding satisfiability. It can be extended to also find the actual satisfiability models for true *QHORN* formulas. As demonstrated in the following section, this can be accomplished in time $O(|\forall| \cdot |\Phi|)$, i.e. without an increase in complexity.

3.7.2. Computing Satisfiability Models for *QHORN* Formulas

The proof of Lemma 3.6.3 describes how a $Z_{\leq 1}$-partial satisfiability model for a formula $\Phi \in$ *QHORN* is obtained by solving $\Phi_{\exists\text{poly}}$. This leads to the following basic algorithm for finding a K_2 satisfiability model for Φ:

1. Transform Φ into $\Phi_{\exists\text{poly}}$ and solve it as discussed above.
2. Obtain a $Z_{\leq 1}$-partial satisfiability model according to the proof of Lemma 3.6.3.
3. Use total completion (Definition 3.3.3) to construct a total satisfiability model.

A closer look at steps 2 and 3 shows that we do not actually have to write down the $Z_{\leq 1}$-partial satisfiability model, because the total completion only needs certain values of the partial model:

$$\begin{aligned} f^t_{y_i}(x_1,...,x_{n_i}) := \quad & (x_1 \vee f_{y_i}(0,1,1,...,1)) \\ \wedge\ & (x_2 \vee f_{y_i}(1,0,1,...,1)) \\ \wedge\ & ... \\ \wedge\ & (x_{n_i} \vee f_{y_i}(1,1,...,1,0)) \\ \wedge\ & f_{y_i}(1,...,1) \end{aligned}$$

In this excerpt from Definition 3.3.3, $f^t_{y_i}$ is the total completion, and f_{y_i} belongs to the partial model. And according to the proof of Lemma 3.6.3, the f_{y_i} are given as

$$\begin{aligned} f_{y_i}(x_1,...,x_{n_i}) = \quad & (\neg x_1 \wedge x_2 \wedge ... \wedge x_{n_i} \ \rightarrow t(y_i^{(0,1,...,1)})) \\ \wedge \ & (x_1 \wedge \neg x_2 \wedge x_3 \wedge ... \wedge x_{n_i} \ \rightarrow t(y_i^{(1,0,1,...,1)})) \\ \wedge \ & ... \\ \wedge \ & (x_1 \wedge ... \wedge x_{n_i-1} \wedge \neg x_{n_i} \ \rightarrow t(y_i^{(1,...,1,0)})) \\ \wedge \ & (x_1 \wedge ... \wedge x_{n_i} \ \rightarrow t(y_i^{(1,...,1)})) \end{aligned}$$

where t is a satisfying truth assignment to the existentials y_i^A (the copies of y_i) in $\Phi_{\exists\text{poly}}$. Now, notice that $f_{y_i}(0,1,...,1) = t(y_i^{(0,1,...,1)})$, etc. That allows us to combine both definitions, and we obtain the following lemma.

Lemma 3.7.1. *Let $\Phi = Q\phi(\mathbf{x},\mathbf{y}) \in QHORN$ be a quantified Horn formula with universal variables $\mathbf{x} = (x_1,...,x_n)$ and existential variables $\mathbf{y} = (y_1,...,y_m)$, and suppose that Φ is satisfiable, which means its expansion $\Phi_{\exists\text{poly}}$ is also satisfiable. Let t be a satisfying truth assignment to the existentials in $\Phi_{\exists\text{poly}}$.*
Then $M = (f_{y_1},..,f_{y_m})$ with

$$\begin{aligned} f_{y_i}(x_1,...,x_{n_i}) := \quad & (x_1 \ \vee t(y_i^{(0,1,1,...,1)})) \\ \wedge \ & (x_2 \ \vee t(y_i^{(1,0,1,...,1)})) \\ \wedge \ & ... \\ \wedge \ & (x_{n_i} \vee t(y_i^{(1,1,...,1,0)})) \\ \wedge \ & t(y_i^{(1,...,1)}) \end{aligned}$$

is a satisfiability model for Φ.

This allows us to refine the algorithm for finding a K_2 satisfiability model for a formula $\Phi \in QHORN$:

1. Solve Φ as in Section 3.7.1 by transforming it into $\Phi_{\exists\text{poly}}$. If $\Phi_{\exists\text{poly}}$ is unsatisfiable, Φ has no satisfiability model. Otherwise, we obtain a satisfying truth assignment to the existentials in $\Phi_{\exists\text{poly}}$.
2. Use this assignment to construct a K_2 satisfiability model according to the previous Lemma 3.7.1.

The first step requires time $O(|\forall| \cdot |\Phi|)$, and the second needs time $O(|\exists| \cdot |\forall|)$. In total, we can find the model in $O(|\forall| \cdot |\Phi| + |\exists| \cdot |\forall|) = O(|\forall| \cdot |\Phi|)$.

Theorem 3.7.2. *Let $\Phi \in QHORN \cap QSAT$ be a satisfiable quantified Horn formula. Then we can find a K_2 satisfiability model for Φ in time $O(|\forall| \cdot |\Phi|)$, where $|\forall|$ is the number of universal quantifiers in Φ, and $|\Phi|$ is the length of Φ.*

This result solves the open question from [KBSZ04] whether there is a quadratic algorithm for computing *QHORN* satisfiability models. It appears that finding the models is just as hard as determining satisfiability.

3.8. Augmenting Propositional Formulas with $QHORN^b$

In Sections 1.1 and 2.6, quantified Boolean formulas with free variables have been motivated as a means to provide (potentially) shorter equivalent representations of propositional formulas. In this scenario, the free variables correspond to the original variables of the propositional formula. Additional variables are introduced as quantified variables, which can, for example, abbreviate repeating parts in the formula, without losing equivalence to the original formula. That input formula may have an arbitrary structure, but the quantified variables are usually added according to a particular algorithm, which means those parts of the resulting QBF^* formula that are formed by the quantified variables may indeed have a special structure.

The following definition introduces such formulas where the quantified variables belong to a particular formula class, while the free variables are in arbitrary *CNF* form.

Definition 3.8.1. *(QK^b)*
Let QK be a subclass of QCNF. With QK^b, we denote all $QCNF^*$ formulas which are in QK after removing literals with free variables.

Proposition 3.8.2. *For any class $QK \subseteq QCNF$, we have the obvious inclusions*

$$QK \subseteq QK^* \subseteq QK^b \ .$$

In Section 3.1, we have already introduced $QHORN^b$ as a special case of the preceding definition. In $QHORN^b$, the Horn property is only enforced on the quantified variables. That means we have a subclass of $QCNF^*$ in which each clause has at most one positive literal over quantified variables, but an arbitrary number of free literals with arbitrary polarity. In Section 2.6, we had the formula

$$\phi(\mathbf{z}) = \bigwedge_{\substack{i,j=1..k,\\ i\neq j}} (\phi_i(\mathbf{z}) \vee \psi_j(\mathbf{z}))$$

where ϕ_i and ψ_j are disjunctions of literals over $\mathbf{z}$. The $QCNF^*$ equivalent

$$\Phi(\mathbf{z}) = \forall x_1 ... \forall x_k \exists y \bigwedge_{i=1..k} ((\neg y \vee \phi_i(\mathbf{z}) \vee x_i) \wedge (y \vee \psi_i(\mathbf{z}) \vee \neg x_i))$$

is in $QHORN^b$, because

$$\Phi^b = \forall x_1 .. \forall x_k \exists y \bigwedge_{i=1..k} ((\neg y \vee x_i) \wedge (y \vee \neg x_i))$$

is a *QHORN* formula. The example illustrates the benefits of going from *CNF* to $QHORN^b$ for formula compression, in this case especially if ϕ_i and ψ_j are long and/or k is large. Interestingly, there are other important applications that fall into the class $QHORN^b$. We will now see that the well-known Tseitin transformation also produces such formulas.

3.8.1. The Tseitin Transformation

The Tseitin procedure (initially suggested in [Tse70]) is used to transform arbitrary propositional formulas in *NNF* into satisfiability-equivalent *CNF* formulas with only polynomial growth. This is a very important task, since SAT solvers or propositional proof systems have often been designed specifically for formulas in *CNF*. The idea of the procedure is to successively replace subformulas $\alpha \vee (\beta \wedge \gamma)$ with $(\alpha \vee \neg y) \wedge (y \vee \beta) \wedge (y \vee \gamma)$ for a new variable y. This can be understood as introducing an abbreviation $\neg y \leftrightarrow (\beta \wedge \gamma)$, which can be simplified to $\neg y \rightarrow (\beta \wedge \gamma)$ without altering the satisfiability of the formula [PG86] [2]. The process continues as long as such subformulas exist. The resulting formula ψ is

[2] We could also abbreviate $y \rightarrow (\beta \wedge \gamma)$, which appears more natural, but that requires negations in two clauses $(\neg y \vee \beta)$ and $(\neg y \vee \gamma)$, instead of only one negation in the clause $(\alpha \vee \neg y)$.

satisfiable if and only if the initial formula ϕ is satisfiable, so we have $\phi \approx_{SAT} \psi$, and it can be shown that the length of ψ is linear in the length of ϕ [KBL99].

In QBF^*, we have a stronger relationship with full equivalence $\alpha \vee (\beta \wedge \gamma) \approx \exists y\, (\alpha \vee \neg y) \wedge (\beta \vee y) \wedge (\gamma \vee y)$. Let ϕ be the initial propositional formula and ψ the output of the Tseitin procedure with newly introduced variables $y_1, \ldots, y_m$. Then we define $\Psi := \exists y_1 \ldots \exists y_m\ \psi$, such that all Tseitin variables are existentially quantified and the original variables in ψ are the free variables of Ψ, and we have full equivalence $\phi \approx \Psi$. Another advantage of this quantified notation is that it becomes immediately clear which variables are the auxiliary variables.

Notice that the Tseitin procedure introduces only existentially quantified variables. With $\exists HORN^b \subseteq QHORN^b$, we denote $QCNF^*$ formulas where an existentially quantified Horn formula is left when the free variables are removed.

Lemma 3.8.3. *The Tseitin transformation produces $\exists HORN^b$ formulas.*

Proof:
The algorithm successively replaces subformulas $\alpha \vee (\beta \wedge \gamma)$. This can be done in a top-down fashion, starting with the outermost misplaced $\wedge$ symbol. A conjunction is misplaced if it has a $\vee$-node as ancestor in the formula tree. After adding new clauses of the form $(\alpha \vee \neg y)$ or $(\beta \vee y)$ in a recursion step, further recursion on $(\alpha \vee \neg y)$ is harmless at this point to the Horn property, because y occurs negatively. On the other hand, if we need to recursively work on a clause $(\beta \vee y)$ which already has a positive existential variable y, all new existentials which are going to be added during the recursion will only occur negatively in clauses with y. Let $\beta = \tilde{\beta}_1 \vee (\tilde{\beta}_2 \wedge \tilde{\beta}_3)$ ($\tilde{\beta}_1$ may also be empty, i.e. $\tilde{\beta}_1 = 0$), then recursion produces new clauses of the form $(\tilde{\beta}_1 \vee y \vee \neg\tilde{y}) \wedge (\tilde{\beta}_2 \vee \tilde{y}) \wedge (\tilde{\beta}_3 \vee \tilde{y})$ when we require that the Tseitin procedure uses a new name for each additional existential variable. Now we can inductively apply this argument to the clauses $(\tilde{\beta}_1 \vee y \vee \neg\tilde{y})$, $(\tilde{\beta}_2 \vee \tilde{y})$ and $(\tilde{\beta}_3 \vee \tilde{y})$. □

3.8.2. Graph Encodings

It is possible to represent propositional formulas as graphs with special structure, a well-known example being Boolean circuits. Such graphs can then be encoded into short equivalent formulas in $\exists HORN^b$, and thus with a matrix in *CNF*, which makes this approach an alternative to the Tseitin procedure from the last section.

A Boolean circuit over $\{\wedge, \vee, \neg\}$ is a directed acyclic graph whose inner nodes are labeled with $\wedge, \vee$, or $\neg$. It has also some source nodes labeled with literals as inputs and exactly one outgoing edge (the sink) to provide an output. Circuits in negation normal form (sometimes also called standard form) have the additional restriction that the negation $\neg$ may not occur as an inner node, only the inputs can be literals x or $\neg x$. By De Morgan's laws, arbitrary circuits can be transformed in linear time into equivalent circuits in negation normal form, so we assume in the following discussion without loss of generality that all circuits are in negation normal form.

It is well known [BBF$^+$73, AB81] that we can associate to any circuit C in negation normal form a satisfiability-equivalent formula in *3-CNF* whose length is linear in the size of the circuit:
First, we label any edge with a new variable. Say the set of these variables is $\{y_1, \ldots, y_m, y\}$, where y denotes the sink.
For a $\wedge$-node $\xrightarrow{y_i} \wedge \xrightarrow{y_j} = \xrightarrow{y_r}$, we obtain the clauses $y_i \to y_r$ and $y_j \to y_r$.
For a $\vee$-node $\xrightarrow{y_i} \vee \xrightarrow{y_j} = \xrightarrow{y_r}$, we obtain the clause $(y_i \wedge y_j) \to y_r$.
For an input edge $L \xrightarrow{y_i}$ with a literal L, we add the clause $L \vee y_i$.
Finally, the sink $\xrightarrow{y}$ is represented by the unit clause $\neg y$.
The matrix of the resulting formula is the conjunction of these clauses. When the variables $y, y_1, ..., y_m$ are quantified existentially, the circuit is logically equivalent to the formula:

$$\begin{aligned}\Phi_C := \exists y \exists y_1 ... \exists y_m \quad & \bigwedge_{\xrightarrow{y_i} \wedge \xrightarrow{y_j} = \xrightarrow{y_r}} (y_i \to y_r) \wedge (y_j \to y_r) \\ \wedge \; & \bigwedge_{\xrightarrow{y_i} \vee \xrightarrow{y_j} = \xrightarrow{y_r}} ((y_i \wedge y_j) \to y_r) \\ \wedge \; & \bigwedge_{L \xrightarrow{y_i}} (L \vee y_i) \wedge \neg y\end{aligned}$$

It is easy to see that this formula is in $\exists HORN^b$ and can be constructed in polynomial time. So we have $CIRCUITS \leq_{poly-time} \exists HORN^b$, and that also implies $PROP \leq_{poly-time} \exists HORN^b$. Notice, however, that propositional formulas only correspond to a special class of circuits in which every inner node has exactly one outgoing edge. Such circuits are said to have fan-out 1. On the other hand, circuits with fan-out greater than 1 have the powerful capability of reusing intermediate results computed by some part of the circuit in multiple subsequent computations in other parts of the circuit. This is similar to the ability of introducing abbreviations by quantified variables in $\exists HORN^b$. Accordingly, the

above transformation of circuits into $\exists HORN^b$ formulas also works without any problems for circuits having fan-out greater than 1. In fact, it is also possible to perform a polynomial-time transformation in the other direction, which implies $CIRCUITS =_{poly-time} \exists HORN^b$ [AB81, KBZB09]. It is widely assumed that circuits with fan-out greater than 1 do not have equivalent polynomial size circuits with fan-out 1, which would also imply that $\exists HORN^b$ is exponentially more powerful than *PROP*, that means $PROP <_{poly-length} \exists HORN^b$. As mentioned in Section 3.2, it is already clear that $CNF <_{poly-length} \exists HORN^b$ [KBL99].

In the encoding Φ_C from above, we observe that the input literals of the circuit only occur in one clause each. All other clauses contain only the existentially quantified auxiliary variables, and quite a lot of these extra variables are needed: one for every edge in the circuit. Using this approach for *CNF* transformation clearly requires more auxiliary variables than the Tseitin transformation, which raises the question whether we can do better than that, and maybe even better than the Tseitin procedure.

We now introduce a new graph-based approach which requires less auxiliary variables. The underlying structure is a multigraph with serial and parallel connections, known as *series-parallel graph* [Sha38, Mac92]. It is not difficult to see that these graphs have the same expressive power as propositional formulas in *NNF* when the edges are labeled with propositional literals [Sha38]. Shannon considers circuits of relays and switches in which he equates serial connections with disjunctions and parallel edges with conjunctions. But mapping given graphs in this way to formulas does obviously not produce *CNF* if a parallel connection is embedded in a serial one. We suggest an extended graph representation from which we can always extract in a straightforward way formulas in $\exists HORN^b$, and thus with a matrix in *CNF*. The idea is to label also the vertices of the graph, but only with new auxiliary variables, which will later be quantified existentially. Furthermore, we extend the labeling of the edges from single input literals to propositional formulas over input literals, in order to make better use of the capability of $\exists HORN^b$ clauses to contain arbitrary numbers of positive or negative free literals. We then write $(u \to w : \alpha)$ for an edge from u to w labeled with a propositional formula α, and we encode it as $\exists u \exists w\ (u \to w) \vee \alpha \approx \exists u \exists w\ \neg u \vee w \vee \alpha$, which can be written as one or more $\exists HORN^b$ clauses (depending on whether α contains conjunctions).

We call the resulting graphs with this labeling scheme for edges and vertices *propositionally-labeled series-parallel graphs* (*PS-graphs*). In order to ensure

the uniqueness of the auxiliary variables, we must supplement the inductive definition of ordinary (unlabeled) series-parallel graphs with additional restrictions on the labels. In the following definition, $vars(E) := \{vars(\alpha) \mid (u \to v : \alpha) \in E\}$ is the set of variables in the labels of an edge set E.

Definition 3.8.4. *(PS-graph)*

1. *Let $V = \{x,y\}$ be a set of nodes, and let $E = \{(x \to y : \alpha)\}$ be a single edge from x to y labeled with a propositional formula α $(x,y \notin vars(\alpha))$. Then $G = (V,E)$ is a PS-graph.*
2. *Let $G_1 = (V_1,E_1)$ and $G_2 = (V_2,E_2)$ be PS-graphs that share the same source x and sink y, with $V_1 \cap V_2 = \{x,y\}$ and $vars(E_1 \cup E_2) \cap (V_1 \cup V_2) = \emptyset$. Then $G = (V_1 \cup V_2, E_1 \cup E_2)$ is a PS-graph.*
3. *Let $G_1 = (V_1,E_1)$ and $G_2 = (V_2,E_2)$ be PS-graphs, such that z is the sink of G_1 and simultaneously the source of G_2, and let $V_1 \cap V_2 = \{z\}$ and $vars(E_1 \cup E_2) \cap (V_1 \cup V_2) = \emptyset$.*
   Then $G = (V_1 \cup V_2, E_1 \cup E_2)$ is a PS-graph.
4. *Only graphs given by the above rules are PS-graphs.*

An example of a well-formed PS-graph is shown in Figure 3.6.

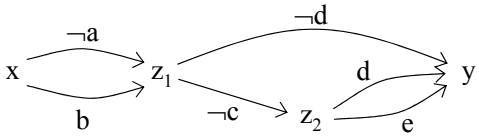

Figure 3.6.: Example of a PS-graph

The semantics of PS-graphs is given implicitly by associating with every PS-graph G a propositional formula Φ_G.

Definition 3.8.5. *(PS-graph Semantics)*
Let $G = (V,E)$ be a PS-graph with source x, sink y and inner nodes $\mathbf{z} = z_1,...,z_t$. Then we associate with G the following formula:

$$\Phi_G = \exists x \exists y \exists z_1 ... \exists z_t \; x \wedge \neg y \wedge \bigwedge_{(u \to w : \alpha) \in E} ((u \to w) \vee \alpha)$$

As mentioned earlier, Φ_G contains a conjunctive term $(u \to w) \vee \alpha$ for every edge $(u \to w : \alpha)$, and two additional unit clauses represent source and sink. If every label α is a literal or a disjunction of literals, we have $\Phi_G \in \exists HORN^b$. If some labels α are more complex and also contain conjunctions, Φ_G can be rewritten into a $\exists HORN^b$ formula. A close look at Φ_G reveals that there are only at most two bound literals per clause, which means the class $\exists 2\text{-}HORN^b \subseteq \exists HORN^b$ is actually sufficient for this application.

For the graph in Figure 3.6, we obtain the following associated $\exists 2\text{-}HORN^b$ formula:

$$\begin{aligned}\Phi_G \;=\; & \exists x \exists y \exists z_1 \exists z_2 \;\; x \wedge \neg y \wedge (\neg x \vee z_1 \vee \neg a) \wedge (\neg x \vee z_1 \vee b) \wedge (\neg z_1 \vee y \vee \neg d) \\ & \qquad\qquad \wedge (\neg z_1 \vee z_2 \vee \neg c) \wedge (\neg z_2 \vee y \vee d) \wedge (\neg z_2 \vee y \vee e) \\ \approx \; & \exists z_1 \exists z_2 \qquad (z_1 \vee \neg a) \wedge (z_1 \vee b) \wedge (\neg z_1 \vee \neg d) \wedge (\neg z_1 \vee z_2 \vee \neg c) \\ & \qquad\qquad \wedge (\neg z_2 \vee d) \wedge (\neg z_2 \vee e)\end{aligned}$$

Interestingly, the semantics definition above coincides with two different intuitive interpretations of PS-graphs. The first one considers all possible paths from the source to the sink and takes the disjunction of labels on such a path.

Theorem 3.8.6. *(Path Semantics of PS-graphs)*
Let $G = (V,E)$ be a PS-graph with source x and sink y, and let Φ_G be the associated formula. Then

$$\Phi_G \approx \bigwedge_{p \text{ path from } x \text{ to } y} \left(\bigvee_{(u \to w : \alpha) \in p} \alpha \right).$$

Proof:
Let $p = (x \to z_1 : \alpha_1), (z_1 \to z_2 : \alpha_2), ..., (z_{t-1} \to z_t : \alpha_t), (z_t \to y : \alpha_{t+1})$ be a path going from the source x to the sink y. Then Φ_G contains the following clauses: $(x), (\neg x \vee z_1 \vee \alpha_1), (\neg z_1 \vee z_2 \vee \alpha_2), \ldots, (\neg z_{t-1} \vee z_t \vee \alpha_t), (\neg z_t \vee y \vee \alpha_{t+1}), (\neg y)$. Omitting the labels α_i leads to an unsatisfiable formula. Therefore, this set of clauses implies the clause $(\alpha_1 \vee \ldots \vee \alpha_{t+1})$. That shows the direction from left to right. For the other direction, let v be a satisfying truth assignment to the right hand formula. We can also apply this truth assignment to the free variables of Φ_G and simplify the resulting formula Φ_G^*. Suppose Φ_G^* is unsatisfiable. Then there must be a chain $(x \to z_1), (z_1 \to z_2), \ldots, (z_{t-1} \to z_t), (z_t \to y)$ in Φ_G^*, since

$\neg y$ is the only negative clause. But that chain represents a path from the source x to the sink y in contradiction to our assumption that the right hand formula is true. □

The second intuitive interpretation of PS-graphs shows that despite the different labeling and the introduction of quantifiers, the role of the parallel and serial connections remains the same as in Shannon's aforementioned series-parallel circuits [Sha38]: parallel connections in the PS-graph can be considered as AND operations, and serial connections can be understood as OR operations.

Theorem 3.8.7. *(AND-OR Semantics of PS-graphs)*
Let $G = (V_1 \cup V_2, E_1 \cup E_2)$ be a PS-graph with source x and sink y that is composed of two smaller PS-graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ with associated formulas $\Phi_{G_1} = \exists \phi_{G_1}$ and $\Phi_{G_2} = \exists \phi_{G_2}$.

1. *Let G_1 and G_2 be arranged in parallel connection where both share the same source x and sink y, $V_1 \cap V_2 = \{x, y\}$ and $vars(E_1 \cup E_2) \cap (V_1 \cup V_2) = \emptyset$ as in Definition 3.8.4 (2).*
   Then $\Phi_G \approx \exists\ \phi_{G_1} \wedge \phi_{G_2}$.
2. *Let G_1 and G_2 be arranged in serial connection where z is the sink of G_1 and the source of G_2, with $V_1 \cap V_2 = \{z\}$ and $vars(E_1 \cup E_2) \cap (V_1 \cup V_2) = \emptyset$ as in Definition 3.8.4 (3).*
   Then $\Phi_G \approx \exists\ \phi_{G_1} \vee \phi_{G_2}$.

Proof:

1. In Φ_G, we can duplicate the unit clauses for source and sink and partition the edges into two disjoint sets:

$$\begin{aligned}
\Phi_G &= \exists x \exists y \exists \mathbf{z} \quad x \wedge \neg y \wedge \textstyle\bigwedge_{(u \to w:\alpha) \in E} ((u \to w) \vee \alpha) \\
&\approx \exists x \exists y \exists \mathbf{z} \quad x \wedge \neg y \wedge \textstyle\bigwedge_{(u \to w:\alpha) \in E_1} ((u \to w) \vee \alpha) \\
&\qquad\qquad \wedge\ x \wedge \neg y \wedge \textstyle\bigwedge_{(u \to w:\alpha) \in E_2} ((u \to w) \vee \alpha) \\
&\approx \exists x \exists y \exists \mathbf{z} \quad \phi_{G_1} \wedge \phi_{G_2}
\end{aligned}$$

2. Due to Theorem 3.8.6, we have $\Phi_G \approx \bigwedge_{p \text{ path from } x \text{ to } y} \left(\bigvee_{(u \to w:\alpha) \in p} \alpha \right)$.
   The construction implies that every path from x to y must pass through z. Consider one single path from x to z. If Φ_G is true, one of the labels on

this path is satisfied, or every path from z to y has one satisfied label. Since this applies to each path from x to z, we have the following implication:

$$\Phi_G \Rightarrow \left(\bigwedge_{\substack{p \text{ path} \\ \text{from } x \text{ to } z}} \left(\bigvee_{(u \to w:\alpha) \in p} \alpha \right) \right) \vee \left(\bigwedge_{\substack{p \text{ path} \\ \text{from } z \text{ to } y}} \left(\bigvee_{(u \to w:\alpha) \in p} \alpha \right) \right)$$

The implication in the other direction is obvious: if all paths in one half of the graph are satisfiable then all paths in the whole graph are satisfiable. Thus it follows that both formulas are equivalent. Once again applying Theorem 3.8.6, the right-hand formula is equivalent to $\Phi_{G_1} \vee \Phi_{G_2}$. The claim follows after moving all quantifiers to the front (notice that $(\exists v\ \phi_{G_1}) \vee (\exists v\ \phi_{G_2}) \approx \exists v\ \phi_{G_1} \vee \phi_{G_2}$). □

This theorem covers cases (2) and (3) of the inductive definition of PS-graphs. For case (1) of Definition 3.8.4, it is easy to see that a graph with only one edge $(x \to y : \alpha)$ simply encodes α, because $\Phi_G = \exists x \exists y\, x \wedge \neg y \wedge (\neg x \vee y \vee \alpha) \approx \alpha$.

3.8.3. *3-CNF* Transformation by PS-Graphs

We have seen that $\exists$*HORN*^{b} or $\exists$*2-HORN*^{b} formulas are a natural way to encode special kinds of graphs that are labeled with propositional literals or formulas, such as Boolean circuits or the new class of PS-graphs. As mentioned before, this can be used for *CNF* transformation: for an input formula ψ in *NNF*, we construct a circuit $C = circ(\psi)$ or a PS-graph $G = ps(\psi)$ that represents ψ. Then the associated encodings $\Phi_C \in \exists$*HORN*^{b} or $\Phi_G \in \exists$*2-HORN*^{b} are equivalent to ψ, and their matrices ϕ_C or ϕ_G are in *CNF*. We can in fact immediately obtain *3-CNF*, which is often easier for existing solvers that seem to have more difficulties when the average clause length increases. The Tseitin procedure, on the other hand, must be extended by an additional processing step when *3-CNF* is desired.

It is well known how to represent a propositional formula as a circuit, but we have not yet explained the construction of a PS-graph G for a formula ψ. Our goal is to build a graph in which all edge labels α are just literals, so that all clauses $\neg u \vee w \vee \alpha$ in Φ_G are in *3-CNF*. It turns out that Theorem 3.8.7 is not only useful for interpreting a PS-graph, but it can also help us build such a

graph, because it establishes a correspondence between the propositional operators AND and OR and the structure of the graph. The idea of our top-down approach is as follows: we start with one single edge that is labeled with the whole formula ψ. Depending on whether ψ is a conjunction or a disjunction of subformulas α_1 and α_2, we then split this edge into two parallel or serial edges labeled with α_1 and α_2. The process continues on the newly created edges until all labels are literals. From G, we can then extract the resulting existentially quantified *3-CNF* formula Φ_G. Listing 3.2 provides the complete transformation procedure.

Listing 3.2: The PS-Transform algorithm

Input propositional formula ψ in NNF;

Initialize $G := (V,E) = (\{x,y\},\{(x \to y : \psi)\})$
// Graph with new nodes x and y and one edge labeled with ψ

while (G has an edge $(u \to w : \alpha)$ with a non−literal formula α) {
  if ($\alpha = \alpha_1 \wedge \alpha_2$)
    $E = E \setminus \{(u \to w : \alpha)\} \cup \{u \to w : \alpha_1,\ u \to w : \alpha_2\}$
  else if ($\alpha = \alpha_1 \vee \alpha_2$)
    $E = E \setminus \{(u \to w : \alpha)\} \cup \{u \to z_i : \alpha_1,\ z_i \to w : \alpha_2\}$
    for a new variable z_i;
}

$\Phi_G = \exists x \exists y \exists z_1 ... \exists z_t\ x \wedge \neg y \wedge \bigwedge_{(u \to w:\alpha) \in E} (\neg u \vee w \vee \alpha)$;

Output PS−graph G with associated 3−CNF formula $\Phi_G \approx \psi$.

Consider the example $\psi = \neg a \wedge ((b \wedge \neg c) \vee (d \wedge e))$. Figure 3.7 shows the construction of G, and we obtain the following associated formula:

$$\Phi_G = \exists x \exists y \exists z\ x \wedge \neg y \wedge (\neg x \vee y \vee \neg a) \wedge (\neg x \vee z \vee b) \wedge (\neg x \vee z \vee \neg c) \\ \wedge (\neg z \vee y \vee d) \wedge (\neg z \vee y \vee e)$$

The unit clauses for source and sink can always be propagated. Then our example requires only one helper variable:

$$\Phi_G = \exists z\ \neg a \wedge (z \vee b) \wedge (z \vee \neg c) \wedge (\neg z \vee d) \wedge (\neg z \vee e)$$

On the other hand, the Tseitin procedure needs two quantified variables to deal

with this example:

$$\begin{aligned}\psi &\approx \exists x\, \neg a \wedge (x \vee (d \wedge e)) \wedge (\neg x \vee b) \wedge (\neg x \vee \neg c) \\ &\approx \exists x \exists y\, \neg a \wedge (x \vee y) \wedge (\neg x \vee b) \wedge (\neg x \vee \neg c) \wedge (\neg y \vee d) \wedge (\neg y \vee e)\end{aligned}$$

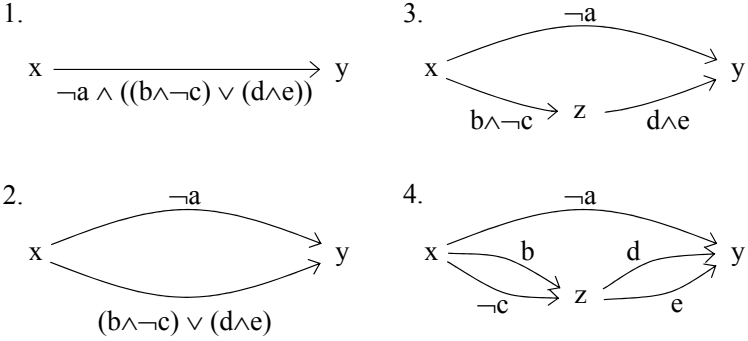

Figure 3.7.: Construction of a PS-graph for $\psi = \neg a \wedge ((b \wedge \neg c) \vee (d \wedge e))$

Theorem 3.8.8. *Let ψ be a propositional formula in NNF which is mapped by the algorithm PS-Transform to the graph $G = ps(\psi)$ with associated formula Φ_G. Then we have $\psi \approx \Phi_G$ and $|\Phi_G| = 3|\psi| + 2$.*

Proof:
The correctness of PS-Transform, that means $\psi \approx \Phi_G$, follows directly from Theorem 3.8.7 and the subsequent remark about single-edge graphs.
To verify that the associated formula Φ_G has linear length, notice that the number of edges in G equals the number of occurrences of literals in the input formula ψ, which is commonly defined to be the length of ψ. Moreover, Φ_G has one 3-clause for each edge in G, plus two unit clauses for source and sink. □

Corollary 3.8.9. *PROP $\leq_{poly-time}$ $\exists$2-HORN^b.*

The algorithm can be modified to produce *CNF* formulas with arbitrary clause length by labeling edges not only with literals, but also with disjunctions of literals. That means we only have to split edges $(u \to w : \alpha)$ if α is neither

a literal nor a disjunction of literals. Or from a bottom-up viewpoint, we can merge two single edges in a serial connection, so that G has no inner nodes for which both indegree and outdegree are 1. With longer clauses, less additional existentially quantified variables are needed: one for each disjunction in the formula tree that has a conjunction as child. This is less than or equal to the number of additional variables that the Tseitin algorithm needs (one for each conjunction that is child of a disjunction). The circuit-based approach that was presented in the introduction clearly needs more additional variables (one for edge in the circuit).

Another advantage of PS-Transform is that it produces $\exists 2\text{-}HORN^b$ formulas, while the other approaches need $\exists HORN^b$ with three bound literals per clause. The Tseitin procedure may actually produce clauses with even more bound literals which must be broken down into clauses with three bound literals in an extra post-processing step that requires additional helper variables. Moreover, PS-Transform guarantees every clause in Φ_G to have at least one literal of the original formula. That should later provide more guidance to solvers or proof systems and avoid extensive reasoning only on helper variables.

Due to the direct correspondence between the propositional connectives and the graph layout, the graph $G = ps(\psi)$ quite naturally represents the structure of the input formula ψ. This is shown by the interesting observation that G encodes both the linear and the exponential *CNF* transformation of ψ. By Theorem 3.8.6, ψ is equivalent to a *CNF* that is obtained by adding for every possible path from the source to the sink a clause which contains the disjunction of labels on such a path. This *CNF* is the result of the application of the distributivity law to ψ.

Consider again the example from Figure 3.7. Here, we have the possible paths $p_1 = (x \to y : \neg a)$, $p_2 = (x \to z : b), (z \to y : d)$, $p_3 = (x \to z : b), (z \to y : e)$, $p_4 = (x \to z : \neg c), (z \to y : d)$ and $p_5 = (x \to z : \neg c), (z \to y : e)$. Then the resulting formula is $\neg a \wedge (b \vee d) \wedge (b \vee e) \wedge (\neg c \vee d) \wedge (\neg c \vee e)$. We can also observe that if a graph $G = ps(\psi)$ has t possible paths from the source to the sink then ψ is equivalent to a *CNF* of at most t clauses.

3.8.4. $QHORN^b$ Complexity Results

We have seen in the previous sections that the class $\exists HORN^b$ is surprisingly powerful. In particular, it can compactly encode circuits with arbitrary fan-out,

which are widely assumed to be exponentially more powerful than circuits with fan-out 1 or, equivalently, propositional logic. We now want to discuss whether we can further increase expressive power by also allowing universal variables.

In the proof of Theorem 3.6.4, we have used arbitrary fixed assignments to the free variables in the $QHORN^*$ formula. A close look at the proof reveals that we have not made any assumptions on the structure of the free variables. In particular, we have not required that the free literals have the Horn property. The proof works exactly the same if we have more than one positive free literal per clause. We can therefore generalize Theorem 3.6.4 and Corollary 3.6.5 from $QHORN^*$ to $QHORN^b$ formulas.

Theorem 3.8.10. *($QHORN^b$ to $\exists HORN^b$ Transformation)*
For any formula $\Phi \in QHORN^b$, there exists an equivalent formula $\Phi' \in \exists HORN^b$ without universal quantifiers. The length of Φ' and the time to compute Φ' from given Φ are both bounded by $|\forall| \cdot |\Phi|$, where $|\forall|$ is the number of universal quantifiers in Φ, and $|\Phi|$ is the length of Φ.

It follows that $QHORN^b =_{poly-time} \exists HORN^b$.

With $PROP \leq_{poly-time} \exists HORN^b$ by one of the transformations from the preceding sections and the observation that a $\exists HORN^b$ formula is satisfiable if and only if its propositional matrix is satisfiable, it is easy to see that the satisfiability problem for $\exists HORN^b$ is *NP*-complete. Because of the previous theorem, the satisfiability problem for $QHORN^b$ is also in *NP*, and its *NP*-completeness follows immediately.

Theorem 3.8.11. *The satisfiability problem for the formula class $QHORN^b$ is NP-complete.*

What this means is that we can take arbitrary propositional *CNF* formulas, for which the satisfiability problem is *NP*-complete, and augment them with existentially and/or universally quantified variables. As long as those quantified variables constitute a *QHORN* formula, the satisfiability problem is not substantially more difficult and remains in *NP*. In this scenario, we receive essentially for free the benefits of introducing quantified variables, such as more compact formula representations (demonstrated by the introductory example at the beginning of Section 3.8) or more natural modeling.

We can also augment non-*CNF* formulas with quantified variables. In that case, we have to be careful to make sure that after the transformation into $QCNF^*$, the quantified variables satisfy the Horn property. Furthermore, Horn-renamable formulas can be detected in linear time [Asp80, Héb94], and renaming quantified variables preserves equivalence. Thus, $ren\text{-}QHORN^b =_{poly-time} QHORN^b$, that means it is also sufficient if the quantified variables are only Horn-renamable.

This positive result is, of course, from a complexity-theoretic viewpoint. Concerning real applications, our results from Chapter 5 indicate that adding large numbers of quantified variables only to achieve the most compact formula representation may in fact cause a considerable overhead and lead to longer solving times. On the other hand, adding some carefully selected quantified variables is indeed very beneficial when it leads to a significant reduction in formula size. It is therefore important to find the right balance between the number of quantified variables and the desired level of formula compression. The benefit of more natural modeling through quantified formulas is apparently immeasurable, but should not be ignored either.

Another point to consider is the following: while we have shown that we can add a *QHORN* component to arbitrary propositional formulas without an increase in complexity, there may in fact be a penalty for augmenting certain subclasses of propositional formulas with *QHORN*. Consider propositional Horn formulas, which can be solved in linear time. Then the combination of *HORN* and *QHORN* is *NP*-complete. This is not caused by the quantifiers, but due to the fact that solving composites of two propositional Horn formulas is already *NP*-complete.

Lemma 3.8.12. *Let $\mathscr{C}$ be the class of CNF formulas $\phi = C_1 \wedge \ldots \wedge C_q$, which are composed of two variable-disjoint propositional Horn formulas $\psi_1, \psi_2 \in HORN$, $\psi_1 = \alpha_1 \wedge \ldots \wedge \alpha_r$ and $\psi_2 = \beta_1 \wedge \ldots \wedge \beta_{r'}$, such that for every clause C_k in ϕ we have $C_k = \alpha_i$ or $C_k = \beta_j$ or $C_k = \alpha_i \vee \beta_j$ for some i, j.*
Then the satisfiability problem for $\mathscr{C}$ is NP-complete.

Proof:
$\mathscr{C}$ satisfiability is obviously in *NP*, because $\mathscr{C} \subseteq PROP$. *NP*-hardness can be shown with a reduction from 3-SAT. Let $\phi \in$ *3-CNF* with $\phi = C_1 \wedge \ldots \wedge C_q$. Then we can determine the satisfiability of ϕ through constructing $\phi' \in \mathscr{C}$ with $\phi' \approx_{SAT} \phi$ by replacing every clause $C_i = (L_{i,1} \vee L_{i,2} \vee L_{i,3})$ in ϕ with new clauses $C'_{i,1} = (a_i \vee L_{i,1})$, $C'_{i,2} = (b_i \vee L_{i,2})$ and $C'_{i,3} = (\neg a_i \vee \neg b_i \vee L_{i,3})$ where a_i and b_i are new variables which do not occur in ϕ. We can easily verify that for every given

value of $L_{i,1},...,L_{i,3}$, it holds that C_i is true $\Leftrightarrow$ $C'_{i,1} \wedge C'_{i,2} \wedge C'_{i,3}$ is satisfiable. It follows that for every truth value assignment to the variables in ϕ, ϕ is true if and only if ϕ' is satisfiable, and therefore $\phi \approx_{SAT} \phi'$. Moreover, $\phi' \in \mathscr{C}$, because ϕ' can be written as the composite (analogous to the definition of $\mathscr{C}$) of $\psi_1 = \bigwedge_{i=1..q}(L_{i,1} \wedge L_{i,2} \wedge L_{i,3})$ and $\psi_2 = \wedge_{i=1..q}(a_i \wedge b_i \wedge (\neg a_i \vee \neg b_i))$. Clearly, $\psi_1, \psi_2 \in HORN$. □

3.9. Summary

This chapter has demonstrated that the syntactic restriction of allowing at most one positive literal per clause influences the semantics of quantified Horn formulas with an interesting effect on the behavior of the quantifiers. We have shown that only cases where at most one of the universally quantified variables is false are relevant for the choice of the existential variables. This has allowed us to provide a detailed characterization of satisfiability models for *QHORN* formulas by focusing only on the relevant parts of the model. Accordingly, the concept of $R_\forall$-partial satisfiability models has been introduced, and it has been shown that for *QHORN* formulas, the partial model can always be extended to a total satisfiability model. We have also investigated models for $QHORN^*$ formulas with free variables and have proved that such equivalence models are monotone.

Based on these results, we have been able to show that

- any formula $\Phi \in QHORN^*$ of length $|\Phi|$ with free variables, $|\forall|$ universal quantifiers and an arbitrary number of existential quantifiers can be transformed into an equivalent quantified Horn formula of length $O(|\forall| \cdot |\Phi|)$ which contains only existential quantifiers.
  It follows that $QHORN^* =_{poly-time} \exists HORN^*$.
- $QHORN^*$-SAT can be solved in time $O(|\forall| \cdot |\Phi|)$ by transforming the input formula into a satisfiability-equivalent propositional formula.
- satisfiability models for *QHORN* formulas can also be computed in time $O(|\forall| \cdot |\Phi|)$.

The above main result that universally quantified Horn formulas have the same expressiveness as purely existential Horn has been extended without much additional effort to $QHORN^b$ formulas with an arbitrary number of positive free

literals per clause. This demonstrates that our approach is quite powerful and naturally captures the structure of Horn formulas.

We have also considered applications of $QHORN^b$ formulas, in particular the transformation of propositional formulas into short equivalent *CNF* formulas by annotating them with quantified variables that obey the Horn property. Using this technique, we have developed a new such transformation that preserves the structure of the input formula in a very natural way. It needs only at most two bound literals per clause, which shows $PROP \leq_{poly-time} \exists 2\text{-}HORN^b$. In addition, the underlying graph representation allows a comprehensible visualization of the algorithm and illustrates a close relationship between exponential CNF transformation by distributing terms and linear CNF transformation with auxiliary variables.

In the following chapter, we will consider more powerful quantifiers which can explicitly indicate variable dependencies. We will see that the Horn property is apparently such a strong restriction that our main observation about the limited impact of universal quantifiers still holds for quantifiers with explicit dependencies.

4. Dependency Quantified Boolean Formulas

Dependency quantified Boolean formulas (*DQBF* or $DQBF^*$ if free variables are allowed) extend quantified Boolean formulas with Henkin-style partially ordered quantifiers. This allows more succinct and more natural formula representations than ordinary QBF^* or even non-prenex QBF^*, which we demonstrate with a new modeling approach for the well-known bounded reachability problem for directed graphs. We compare our encoding with existing QBF^* representations and show that it has a shorter matrix and requires less quantified variables: for a graph with 2^n vertices, we need at most $O(n)$ variables, in contrast to $O(n^2)$ (but in an unbounded number of quantifier blocks) or even $O(2^n)$ in QBF^*.

In general, $DQBF^*$ satisfiability is *NEXPTIME*-complete. Accordingly, we define easier subclasses through restrictions on the structure of the prefix or the clauses. We show that formulas with dependencies of bounded or at most logarithmic size (D_kQBF^* or $D_{log}QBF^*$) have Σ_2^P-complete satisfiability problems. In addition, formulas with polynomially orderable dependencies ($D_{po}QBF^*$) are considered. They are shown to be a *PSPACE*-complete generalization of QBF^*. We also investigate dependency quantified Horn formulas ($DQHORN^*$) and lift from ordinary quantified Horn the characteristic property that the behavior of the existential quantifiers depends only on the cases where at most one of the universal variables is zero. This allows us to solve $DQHORN^*$ formulas Φ with $|\forall|$ universal quantifiers in time $O(|\forall| \cdot |\Phi|)$, which equals the best known algorithms for the $QHORN^*$ satisfiability problem.

An important tool in our investigations is universal quantifier expansion. We prove its correctness for $DQBF^*$ and observe that it can eliminate dependencies and transform $DQBF^*$ formulas into QBF^*. While this may cause exponential formula growth in general, the process can be shown to remain polynomial for $D_{po}QBF^*$ formulas. A quadratic bound can be established for $DQHORN^*$ formulas and their generalization $DQHORN^b$ with an arbitrary number of positive

free literals per clause. It follows that $DQHORN^b$ satisfiability is *NP*-complete and that $DQHORN^b =_{poly-time} QHORN^b =_{poly-time} \exists HORN^b$.

This chapter extends preliminary results on $DQHORN^*$ which have been published in [BKB06].

4.1. Motivation

We have seen that quantification is a powerful tool to provide short representations for propositional formulas, and also more natural encodings for many problems that already have an inherent forall/exists semantics. There is, however, one clear limitation in the concept of quantification which we have encountered so far: in QBF^*, it is not possible to explicitly state on which universals an existential variable depends. This is in particular evident for formulas in prenex form where all quantifiers appear at the beginning and each existential variable depends on all preceding universals.

4.1.1. Variable Dependencies and Formula Structure

It has already been pointed out that quantified Boolean formulas are usually assumed by definition to be in prenex form, and that is also the input format generally required by QBF^* solvers. Prenex formulas can be handled more efficiently because of their clear and simple concept of variable scopes and dependencies, and operations like Q-resolution become difficult to perform otherwise. But this is traded for a loss of structural information. Consider a non-prenex formula of the form

$$\Phi = (\forall x\ \phi(x)) \wedge (\exists y\ \psi(y))$$

which consists of two variable-disjoint subformulas. Then there are two equivalent prenex representations:

$$\Phi_{pre-\forall\exists} = \forall x \exists y\ \phi(x) \wedge \psi(y)$$
$$\Phi_{pre-\exists\forall} = \exists y \forall x\ \phi(x) \wedge \psi(y)$$

From a model-centered viewpoint, the first representation is clearly disadvantageous, because y now seems to depend on x, which means a model-based solver

will expect a unary function and only later discover that a Boolean constant is actually sufficient. Similarly, the second prenex representation might cause a DPLL-based solver to recursively branch on x a second time if the first attempt to assign y is not successful. So in both cases, we lose some of the structural information inherent in the original formula, and that may lead to a larger search space. Recent experimental studies [EST+04, GNT07] have shown that this problem may have a considerable impact on the performance of *QBF** solvers.

How can we overcome the information loss from flattening quantifier hierarchies into prenex form? An obvious solution is to extend *QBF** solvers to directly handle non-prenex formulas [GNT07]. However, non-prenex formulas may have complex nested local quantification scopes which implicitly define the variable dependencies. Accordingly, non-prenex formulas are often difficult to read, and it can be challenging to write correct and concise non-prenex *QBF** encodings of problems. To illustrate the subtleties of non-prenex formulas, consider the following example:

$$\Psi = (\forall a\ \phi_1 \wedge (\forall b\ (\exists c\ \phi_2) \wedge \phi_3) \wedge (\exists d\ \phi_4) \wedge \phi_5) \wedge (\exists e\ \phi_6) \wedge \phi_7$$

On which universals do c, d and e depend?

Another solution which retains the notational simplicity of prenex form is to recover lost information from the formula structure during the solving process. As discussed in more detail in Chapter 5, a solver can analyze the local connectivity of variables in common clauses and infer that variables which occur in disjoint subformulas do not depend on each other.

Both approaches, structure analysis and non-prenex formulas, require that the variable dependencies correspond to the formula structure. This requirement excludes a formula of the form

$$\Gamma = \phi(x_1, x_2, y_1) \wedge \psi(x_1, x_3, y_2) \wedge \omega(x_1, x_2, x_3, y_1, y_2)$$

with universals $x_1, \ldots, x_3$ and existentials y_1, y_2 where y_1 depends on x_1 and x_2, but *not* x_3, and y_2 depends on x_1 and x_3, but *not* x_2. So we have a tree-like quantifier hierarchy with x_1 at the root and y_1 and y_2 as two separate leaves. If there was no subformula ω, we could easily provide a suitable non-prenex formula:

$$\Gamma_{\phi} = \forall x_1\ (\forall x_2 \exists y_1\ \phi(x_1, x_2, y_1)) \wedge (\forall x_3 \exists y_2\ \psi(x_1, x_3, y_2))$$

But if ω is present, the scopes of y_1 and y_2 are not disjoint, which opposes such a non-prenex representation. Right now, it may not yet be clear why it is useful to have expressions like ω which combine existentials from disjoint subformulas. We will later investigate how to use this feature for novel encodings. At this point, we just emphasize that the example represents a class of formulas which are unlikely to have short non-prenex QBF^* representations. This is different from the earlier examples in this section, where the information loss from prenexing has only affected the solving performance, but we have still been able to provide concise non-prenex and prenex QBF^* encodings.

We are therefore going to consider an extension of QBF^* with more expressive quantifiers that can explicitly indicate variable dependencies. This does not only provide us with a clear and intuitive notation, but exceeds the expressive power of prenex and non-prenex QBF^* and allows new modeling approaches where variable dependencies may be independent of the formula structure.

4.1.2. Dependency Quantifiers

Partially-ordered quantifiers with explicit variable dependencies have initially been proposed by Henkin [Hen61] for first-order predicate logic. Such *branching quantifiers*, or simply *Henkin quantifiers*, are typically written as a two-dimensional matrix where each row contains an existentially quantified variable that is preceded by exactly those universals on which it depends. It has been shown in [Wal70] that first-order formulas with Henkin quantifiers contain the class of finite existentially quantified second-order logic. That means the addition of Henkin quantifiers can indeed lead to a clear increase in expressiveness. We want to achieve a similar enhancement with quantified Boolean formulas. If we adopt the notion of Henkin quantifiers to *QBF*, we can correctly encode the previous example as follows:

$$\begin{pmatrix} \forall x_1 \forall x_2 \exists y_1 \\ \forall x_1 \forall x_3 \exists y_2 \end{pmatrix} \phi(x_1, x_2, y_1) \wedge \psi(x_1, x_3, y_2) \wedge \omega(x_1, x_2, x_3, y_1, y_2)$$

The informal semantics is that y_1 depends on x_1 and x_2, and y_2 depends on x_1 and x_3. Since the only relevant information is which universal quantifiers precede which existential quantifier, we can use a (typographically) simpler notation as follows:

$$\forall x_1 \forall x_2 \exists y_1(x_1, x_2) \forall x_3 \exists y_2(x_1, x_3)\ \phi(x_1, x_2, y_1) \wedge \psi(x_1, x_3, y_2) \wedge \omega(x_1, x_2, x_3, y_1, y_2)$$

For each existential quantifier, we indicate the universal variables on which it depends, and we call such a quantifier a *dependency quantifier*. Without loss of information, we can assume that the prefix is always in the form $\forall^*\exists^*$:

$$\forall x_1 \forall x_2 \forall x_3 \exists y_1(x_1,x_2) \exists y_2(x_1,x_3)\, \phi(x_1,x_2,y_1) \wedge \psi(x_1,x_3,y_2) \wedge \omega(x_1,x_2,x_3,y_1,y_2)$$

This notation has been introduced for quantified Boolean formulas by Peterson, Azhar and Reif in [PRA01] (a preliminary version appeared in [PR79]) under the name *Dependency Quantified Boolean Formulas (DQBF)*. It is no coincidence that the syntax of the existential dependency quantifiers resembles the notation of functions, because in both predicate logic and *DQBF*, dependency quantified existential variables can be associated with (Skolem) functions. In the case of *DQBF* formulas, we are going to map the dependency quantified existential variables to satisfiability model functions that take the dominating universals as parameters.

4.2. Research Goals and Related Work

Partially-ordered quantification has been around for quite some time, but has not been widely used in combination with quantified Boolean formulas. This is probably due to the fact that *DQBF* satisfiability is *NEXPTIME*-complete, which has been shown by Peterson, Reif and Azhar [PRA01] immediately upon introducing *DQBF*. Assuming that $NEXPTIME \neq PSPACE$, *NEXPTIME* is a jump in complexity compared to *QBF* satisfiability which is *PSPACE*-complete. In analogy to QBF^*, we also define dependency quantified Boolean formulas with free variables ($DQBF^*$). While these are not considered in [PRA01], it is easy to see that the *NEXPTIME*-completeness also holds for $DQBF^*$ satisfiability.

We first want to investigate which particular feature of dependency quantification is responsible for this increase in complexity, and how to use it to obtain more concise encodings of particular problems. The introduction in Section 4.1.1 suggests that the main advantage of $DQBF^*$ over QBF^* in terms of expressiveness is the ability to have variable dependencies which are independent of the formula structure. That means having existential variables which depend on distinct universals, but still occur in a common subformula. How can we make use of this capability? [BFL91] and [PRA01] have shown that

the class *NEXPTIME* can be characterized by multi-prover interactive proof systems [BOGKW88] or games in which two non-communicating provers or players perform independent computations in order to convince or win against a third entity. Here, the emphasis is on the independence of the two provers or players (if they did cooperate, they could be simulated by one single instance), and that independence can be guaranteed in $DQBF^*$ by encoding them as two sets of existential variables with disjoint dependencies.

We demonstrate that such multi-prover approaches permit better reuse of quantified variables by allowing multiple assignments to the same variable without loss of earlier values. This pattern is then applied to develop a compact $DQBF^*$ encoding of the well-known bounded reachability problem for directed graphs [Pap94] which is fundamental to bounded model checking [BCCZ99, CBRZ01]. Existing QBF^* encodings introduce quantified variables to avoid multiple copies of the graph transition relation. The new encoding that we present can also accomplish this, but it needs less equality checks in the formula matrix, and it requires less quantified variables and/or less quantifier alternations than the QBF^* representations: for a graph with 2^n vertices, e.g. the state space of n binary variables, we only need $O(n)$ quantified variables with a simple $\forall^*\exists^*$ prefix, compared to existing QBF^* encodings with $O(n^2)$ variables in a prefix with $O(n)$ quantifier alternations [Pap94, DHK05] or $O(2^n)$ variables with $\exists^*\forall^*$ prefix [DHK05].

Faced with the *NEXPTIME*-completeness of $DQBF^*$ satisfiability, an important question is whether there are subclasses of $DQBF^*$ that are substantially easier, yet still powerful enough to benefit from dependency quantifiers and to be interesting for applications. Might it even be possible to add dependency quantification to a non-trivial subclass of QBF^* without seeing a hefty increase in complexity? In QBF^*, the two most prominent approaches to obtain easier subclasses are the following:

- restrictions on the structure of the quantifier prefix
- constraints on the clause structure

In this chapter, we demonstrate that interesting $DQBF^*$ subclasses with lower complexity can be obtained by both kinds of restrictions. For the prefix structure, suitable constraints are not immediately obvious: limiting the number of quantifier alternations as it is done in QBF^* is pointless with dependency quantifiers, since they can always be written with a simple $\forall^*\exists^*$ prefix without any

loss in expressiveness. We suggest to consider the structure of the dependency lists instead. An obvious constraint on the dependencies is to limit their sizes, that is, the maximum number of universals on which each existential variable may depend. We show that formulas with dependencies of bounded or at most logarithmic sizes (D_kQBF^* or $D_{log}QBF^*$) have Σ_2^p-complete satisfiability problems. When we consider QBF^* formulas as an embedded subset of $DQBF^*$, we obtain $DQBF^*$ formulas with dependencies that are pairwise included in each other. We show that this class of formulas can be generalized to formulas with dependencies that are not totally ordered, but can be ordered within polynomial space, so that the satisfiability problem remains *PSPACE*-complete.

A main result in this context is that we are able to show that Horn formulas remain polynomially solvable even with dependency quantifiers. This makes $DQHORN^*$ the first non-trivial class of formulas known to be tractable in combination with dependency quantification. The Horn property appears to be such a strong restriction that the observation from Chapter 3 still holds: a linear number of assignments to the universal variables completely determines the behavior of the existentials, even if they are dependency quantified.

To our knowledge, there are no $DQBF^*$ solvers available yet. We suggest that an intermediate step to this ultimate goal might be a system which externally accepts dependency quantified formulas, but internally preprocesses them into QBF^* and feeds them to an encapsulated QBF^* engine. That would give users the benefit of a clearer notation with explicit variable dependencies, which removes the need for non-prenex formulas and allows for more natural modeling of problems, while the system itself could be based on existing QBF^* techniques. We demonstrate that the translation to QBF^* can be accomplished with the universal quantifier expansion method that has already played a prominent role in the previous chapter. For $D_{po}QBF^*$ and $DQHORN^*$ as well as its generalization $DQHORN^b$ with arbitrarily many positive free literals per clause, we show that this transformation requires only polynomial time. Besides the conversion of $DQBF^*$ into QBF^*, universal expansion might also be helpful in building a full $DQBF^*$ solver, considering that expansion is one of the most successful solving techniques in the QBF^* area.

4.3. Fundamentals

Before we can investigate in detail the questions raised in the previous section, we need a more thorough definition of syntax and semantics of *DQBF* formulas that goes beyond the brief informal introduction in [PRA01]. In addition, we introduce *DQBF** formulas with free variables and explain how key concepts like models can be lifted from *QBF* and *QBF**.

4.3.1. *DQBF* Syntax and Semantics

We first introduce a notation which allows us to quickly enumerate the dependencies of a given existential variable y_i in a $\forall^*\exists^*$ prefix with n universal quantifiers and m dependency quantified existentials. We are using indices $d_{i,1},...,d_{i,n_i}$ which point to the n_i universals on which y_i depends. For example, given the existential quantifier $\exists y_4(x_3,x_5)$, we say that y_4 depends on $x_{d_{4,1}}$ and $x_{d_{4,2}}$ with $d_{4,1}=3$ and $d_{4,2}=5$. To avoid confusion, we require $1 \leq d_{i,j} \leq n$, so that the dependency lists only reference universals which have been introduced by a preceding universal quantifier. Moreover, we assume that $d_{i,j} \neq d_{i,k}$ for $j \neq k$, because it makes no sense to have duplicate entries of a universal variable in the same dependency list. We also allow empty dependencies with $n_i = 0$, i.e. existential quantifiers $\exists y_i()$ that do not depend on any universals.

Definition 4.3.1. *(Dependency Quantified Boolean Formula)*
*A **dependency quantified Boolean formula** $\Phi \in DQBF$ with universal variables $\mathbf{x} = (x_1,...,x_n)$ and existential variables $\mathbf{y} = (y_1,...,y_m)$ $(n,m \geq 0)$ is a formula of the form*

$$\Phi = \forall x_1 ... \forall x_n \exists y_1(x_{d_{1,1}},...,x_{d_{1,n_1}}) ... \exists y_m(x_{d_{m,1}},...,x_{d_{m,n_m}})\ \phi(\mathbf{x},\mathbf{y})$$

where the matrix ϕ is a propositional formula over the quantified variables.

We often use a shorter notation $\Phi = \forall x_1 ... \forall x_n \exists y_1(\mathbf{x_{d_1}}) ... \exists y_m(\mathbf{x_{d_m}})\ \phi(\mathbf{x},\mathbf{y})$ where we abbreviate $\mathbf{x_{d_i}} := (x_{d_{i,1}},...,x_{d_{i,n_i}})$. Alternatively, we also treat dependency lists $\mathbf{x_{d_i}}$ as sets and apply the basic set operations and relations on them. That allows us to write expressions like $x_1 \in \mathbf{x_{d_i}}$ or $\mathbf{x_{d_i}} \subseteq \mathbf{x_{d_j}}$. Furthermore, we sometimes use the shorthand $\exists y_i, y_j(\mathbf{x_d}) := \exists y_i(\mathbf{x_d}) \exists y_j(\mathbf{x_d})$ in order to combine existential variables with the same dependencies.

The definition requires that *DQBF* formulas are in prenex form. This avoids having negations of existential dependency quantifiers, which would otherwise be problematic, because the quantifier inversion rule $\neg(\exists y\ \phi) \approx \forall y\ \neg\phi$ which is well known for *QBF* does not work in general for existential dependency quantifiers. Consider the following simple example:

$$\Psi = \forall x_1 \forall x_2 \exists y(x_1)\ (x_2 \vee y) \wedge (\neg x_2 \vee \neg y)$$

Then Ψ is clearly false, because y cannot adapt to different values of x_2. But

$$\neg\Psi = \neg\left(\forall x_1 \forall x_2 \exists y(x_1)\ (x_2 \vee y) \wedge (\neg x_2 \vee \neg y)\right) \not\approx \exists x_1 \exists x_2 \forall y\ (\neg x_2 \wedge \neg y) \vee (x_2 \wedge y)$$

because the formula on the right hand side is obviously false as well. The problem is that the dual $\neg DQ\ \phi$ of a dependency quantifier DQ can usually not be expressed as an ordinary non-negated dependency quantifier. The issue is the same for first-order logic with Henkin quantifiers and is discussed in more detail in [BG86]. But since dependency quantifiers are expressive enough to simulate the local quantifier scopes in a non-prenex formula, there is not much sense in allowing hard-to-read non-prenex formulas anyway. All the information contained in the nesting of quantifier scopes can be encoded explicitly in the variable dependencies. This also allows us to assume without loss of generality that the prefix is always in the form $\forall^*\exists^*$ as in the definition above.

So far, we have discussed dependency quantifiers with an informal understanding of their semantics. Formally, we define the semantics of *DQBF* over model functions. Satisfiability models for *DQBF* formulas are a straightforward generalization of the *QBF* case (Definition 2.7.1). In *DQBF*, each model function f_{y_i} contains exactly those universals $x_{d_{i,1}}, ..., x_{d_{i,n_i}}$ on which y_i depends.

Definition 4.3.2. *(DQBF Satisfiability Model)*
*For $\Phi \in DQBF$ with existential variables $\mathbf{y} = (y_1, ..., y_m)$, let $M = (f_{y_1}, ..., f_{y_m})$ be a mapping which associates with each existential variable y_i a propositional formula f_{y_i} over the universal variables $x_{d_{i,1}}, ..., x_{d_{i,n_i}}$ on which y_i depends. Then M is a **satisfiability model** for Φ if the purely universally quantified QBF formula $\Phi[\mathbf{y}/M] := \Phi[y_1/f_{y_1}, ..., y_m/f_{y_m}]$, where simultaneously each existential variable y_i is replaced by its corresponding formula f_{y_i} and the existential quantifiers are dropped from the prefix, is true.*

Definition 4.3.3. *(DQBF Semantics)*
*A DQBF formula is **true** if and only if it has a satisfiability model.*

For *QBF*, the last definition is actually a theorem, because its semantics is usually defined inductively without referring to model functions. That is not feasible for *DQBF*, because quantifiers in *DQBF* prefixes cannot be eliminated from outermost to innermost by considering only one quantifier at a time. Instead, the dependencies require considering cross-references between quantifiers, which is problematic for an inductive definition.

4.3.2. The Class $DQBF^*$ with Free Variables

So far, we have only considered closed *DQBF* formulas. We can also allow free variables and define the class $DQBF^*$ as follows:

Definition 4.3.4. *($DQBF^*$ Syntax, Satisfiability, Equivalence and Entailment) A **dependency quantified Boolean formula** $\Phi(\mathbf{z}) \in DQBF^*$ **with free variables** $\mathbf{z} = (z_1, ..., z_r)$ is a formula of the form*

$$\Phi(\mathbf{z}) = \forall x_1 ... \forall x_n \exists y_1(x_{d_{1,1}}, ..., x_{d_{1,n_1}}) ... \exists y_m(x_{d_{m,1}}, ..., x_{d_{m,n_m}})\ \phi(\mathbf{x}, \mathbf{y}, \mathbf{z})$$

with universal variables $\mathbf{x} = (x_1, ..., x_n)$, existential variables $\mathbf{y} = (y_1, ..., y_m)$ $(n, m \geq 0)$ and the matrix ϕ given by a propositional formula over the quantified and the free variables.

*A $DQBF^*$ formula $\Phi(\mathbf{z})$ is **satisfiable** if and only if there exists a truth assignment $\tau(\mathbf{z}) = (\tau(z_1), ..., \tau(z_r)) \in \{0,1\}^r$ to the free variables such that*

$$\Phi(\tau(\mathbf{z})) = \forall x_1 ... \forall x_n \exists y_1(\mathbf{x}_{\mathbf{d}_1}) ... \exists y_m(\mathbf{x}_{\mathbf{d}_m})\ \phi(\mathbf{x}, \mathbf{y}, \tau(\mathbf{z}))$$

is true.

*Two $DQBF^*$ formulas $\Psi_1(z_1, ..., z_r)$ and $\Psi_2(z_1, ..., z_r)$ are **equivalent** ($\Psi_1 \approx \Psi_2$) if and only if $\Psi_1 \models \Psi_2$ as well as $\Psi_2 \models \Psi_1$, where **semantic entailment** $\models$ is defined as follows: $\Psi_1 \models \Psi_2$ if and only if for all truth value assignments $t(\mathbf{z}) = (t(z_1), ..., t(z_r)) \in \{0,1\}^r$ to the free variables, we have $\Psi_1(t(\mathbf{z})) = 1 \Rightarrow \Psi_2(t(\mathbf{z})) = 1$.*

We can see that $DQBF^*$ satisfiability, equivalence and entailment are analogous to the corresponding definitions for QBF^*. In fact, we can map any QBF^* formula to a corresponding $DQBF^*$ formula with the same matrix and the same

variables such that the dependency list of each existential variable contains exactly those universal variables that precede the existential in the QBF^* prefix. On this embedding of QBF^* into $DQBF^*$, basic concepts like satisfiability and equivalence just coincide with the corresponding QBF^* definitions.

Notice that in the above definition of $DQBF^*$ satisfiability, the truth assignment to the free variables is determined before the values of the existential variables are chosen. That means an existentially quantified variable y_i actually depends not only on the explicitly indicated universal variables $x_{d_{i,1}},..,x_{d_{i,n_i}}$, but also on all free variables - even though these are not (and cannot be) contained in the dependency list of y_i.

It might seem natural to add to the language the ability to also explicitly indicate on which free variables an existential variable depends, but that would require substantial modifications to the definition of $DQBF^*$, so that we would lose the close similarity to QBF^*, where an existentially quantified variable also depends on all free variables, regardless of its actual scope. Furthermore, our approach maintains the very useful property that a $DQBF^*$ formula can be considered as a closed $DQBF$ formula when the free variables are fixed with a given truth assignment.

The discussion from the last paragraph suggests that we should describe an existential variable in a $DQBF^*$ formula by a function over the free variables and over those universals on which it depends. That leads to the following definition of $DQBF^*$ equivalence models as a generalization of QBF^* equivalence models (Definition 2.7.2):

Definition 4.3.5. *($DQBF^*$ Equivalence Model)*
Let $\Phi(\mathbf{z}) = \forall x_1...\forall x_n \exists y_1(\mathbf{x_{d_1}})...\exists y_m(\mathbf{x_{d_m}})\ \phi(\mathbf{x},\mathbf{y},\mathbf{z})$ *be a* $DQBF^*$ *formula. For propositional formulas* f_{y_i} *over* $\mathbf{z}$ *and over universals* $\mathbf{x_{d_i}} = (x_{d_{i,1}},...,x_{d_{i,n_i}})$ *on which* y_i *depends,* $M = (f_{y_1},...,f_{y_m})$ *is an* ***equivalence model*** *for* $\Phi(\mathbf{z})$ *if and only if* $\Phi(\mathbf{z}) \approx \Phi(\mathbf{z})[\mathbf{y}/M] = \forall x_1...\forall x_n\ \phi(x_1,...,x_n, f_{y_1}(\mathbf{z},\mathbf{x_{d_1}}),...,f_{y_m}(\mathbf{z},\mathbf{x_{d_m}}),\mathbf{z})$.

4.3.3. $DQBF^*$ Complexity and Expressiveness

While $DQBF^*$ formulas are not considered in [PRA01], it is easy to see that the proof of the *NEXPTIME*-completeness of $DQBF$ satisfiability can be lifted to $DQBF^*$: the *NEXPTIME*-hardness of $DQBF$ automatically extends to its superclass $DQBF^*$, so it only remains to verify that $DQBF^*$ satisfiability is also

in *NEXPTIME*. Given $\Phi \in DQBF^*$, we first guess a satisfying truth assignment $\tau(\mathbf{z})$ to the free variables and replace all occurrences of free variables z_i with the corresponding truth value $\tau(z_i)$. For the resulting *DQBF* formula $\Phi(\tau(\mathbf{z}))$, we then guess a satisfiability model $M = (f_{y_1}, ..., f_{y_m})$. Both steps can be accomplished in nondeterministic exponential time. Then we only have to verify that the matrix ϕ is true for every assignment to the universal variables $\mathbf{x}$ when the free variables are assigned as in $\tau(\mathbf{z})$ and the value of each existential variable y_j is looked up in M for the given values of $\mathbf{x}_{\mathbf{d}_j}$ and $\mathbf{z}$. This is clearly possible in exponential time.

Theorem 4.3.6. *The satisfiability problem for $DQBF^*$ is NEXPTIME-complete.*

A $DQBF^*$ formula can be converted into QBF^* by eliminating the existential variables, because a purely universally quantified $DQBF^*$ formula has no dependencies. Existential variables can be eliminated by replacing them with the corresponding equivalence model functions: for $\Phi \in DQBF^*$ with existential variables $\mathbf{y}$, we have $\Phi(\mathbf{z}) \approx \Phi(\mathbf{z})[\mathbf{y}/M] \in QBF^*$ with an equivalence model M. The following theorem shows that this is always possible, because each $DQBF^*$ formula has such an equivalence model M.

Theorem 4.3.7. *Every $DQBF^*$ formula $\Phi(\mathbf{z})$ has an equivalence model M and is therefore equivalent to a universally quantified QBF^* formula $\Phi(\mathbf{z})[\mathbf{y}/M]$.*

Proof:
Let $\Phi(\mathbf{z}) = \forall x_1 ... \forall x_n \exists y_1(x_{d_{1,1}}, ..., x_{d_{1,n_1}}) ... \exists y_m(x_{d_{m,1}}, ..., x_{d_{m,n_m}})\ \phi(\mathbf{x}, \mathbf{y}, \mathbf{z})$ be in $DQBF^*$ with universal variables $\mathbf{x} = x_1, ..., x_n$, existentials $\mathbf{y} = y_1, ..., y_m$ and free variables $\mathbf{z} = z_1, ..., z_r$. Furthermore, let $t(\mathbf{z}) = (t(z_1), ..., t(z_r))$ be a truth assignment to the free variables. If the closed formula $\Phi(t(\mathbf{z}))$ is true, it has a satisfiability model $M^{(t(\mathbf{z}))} = (f_{y_1}^{(t(\mathbf{z}))}, ..., f_{y_m}^{(t(\mathbf{z}))})$, otherwise let $f_{y_i}^{(t(\mathbf{z}))}(x_{d_{i,1}}, ..., x_{d_{i,n_i}}) := 0$ for $i = 1, ..., m$. We label the model functions with the corresponding assignment $t(\mathbf{z})$, so that we can assemble them into equivalence model functions for arbitrary values of $\mathbf{z}$:

$$f_{y_i}(z_1, ..., z_r, \mathbf{x}_{\mathbf{d}_i}) := \begin{array}{l} \Big((\neg z_1 \wedge \neg z_2 \wedge ... \wedge \neg z_r) \rightarrow f_{y_i}^{(0,...,0)}(\mathbf{x}_{\mathbf{d}_i})\Big) \wedge \\ \Big((z_1 \wedge \neg z_2 \wedge ... \wedge \neg z_r) \rightarrow f_{y_i}^{(1,0,...,0)}(\mathbf{x}_{\mathbf{d}_i})\Big) \wedge \\ \qquad \vdots \\ \Big((z_1 \wedge z_2 \wedge ... \wedge z_r) \rightarrow f_{y_i}^{(1,...,1)}(\mathbf{x}_{\mathbf{d}_i})\Big) \end{array}$$

Then let $M = (f_{y_1}, ..., f_{y_m})$, and for any truth assignment $t(\mathbf{z})$, it follows that if $\Phi(t(\mathbf{z}))$ is true, $\Phi(t(\mathbf{z}))[\mathbf{y}/M]$ is also true, because $f_{y_i}(t(\mathbf{z}), \mathbf{x}_{\mathbf{d}_i}) = f_{y_i}^{(t(\mathbf{z}))}(\mathbf{x}_{\mathbf{d}_i})$ and $M^{(t(\mathbf{z}))}$ is a satisfiability model. And if $\Phi(t(\mathbf{z}))$ is false, it is also false in the special case $\Phi(t(\mathbf{z}))[\mathbf{y}/M]$ when all existentials are set to 0. It follows that $\Phi(\mathbf{z}) \approx \Phi(\mathbf{z})[\mathbf{y}/M]$. □

If *NEXPTIME* ≠ *PSPACE*, there are *DQBF** formulas $\Phi(\mathbf{z})$ for which the resulting formula $\Phi(\mathbf{z})[\mathbf{y}/M]$ must be exponentially larger, which means M must have exponential size. If that were not the case, we could guess in nondeterministic polynomial space (*NPSPACE* = *PSPACE*) the necessary equivalence model functions, substitute them into the formula and solve the resulting *QBF** deterministically in polynomial space.

While dependency quantifiers appear to be a powerful language feature, there are, however, Boolean functions for which even *DQBF** cannot provide succinct representations.

Theorem 4.3.8. *There is no polynomial p such that for every Boolean function $f : \{z_1, ..., z_r\} \to \{0,1\}$ there exists a dependency quantified Boolean formula $\Phi_f \in$ DQBF* which has $z_1, ..., z_r$ as free variables and for which $f \approx \Phi_f$ and $|\Phi_f| \leq p(r)$.*

Proof:
Without loss of generality, we can assume that Φ_f is of the following form:

$$\Phi_f(z_1, ..., z_r) = \forall x_1 ... \forall x_n \exists y_1(x_{d_{1,1}}, ..., x_{d_{1,n_1}}) ... \exists y_m(x_{d_{m,1}}, ..., x_{d_{m,n_m}})\ \phi(\mathbf{x}, \mathbf{y}, \mathbf{z})$$

A formula of length $p(r)$ contains at most $p(r)$ different variables. We now consider the different components of the formula:

- $\forall^*$*-part of the prefix:* there are at most $p(r)^n$ different $\forall^*$-blocks.
- $\exists^*$*-part of the prefix:* we can safely assume that $n_i = n$: if a variable y_i depends on less than n universals, we can pad the list of dependencies with $x_{d_{i,1}}$. Then there are at most $(p(r) \cdot p(r)^n)^m = p(r)^{(n+1)m}$ different $\exists^*$-blocks.
- *formula matrix:* as discussed in [KBL99], there are at most $(p(r)+3)^{4p(r)}$ different propositional formulas of length $p(r)$.

With $n, m \leq p(r)$, we can combine the three cases and obtain the following upper bound for the number of distinct $DQBF^*$ formulas of length $p(r)$:

$$p(r)^{p(r)} \cdot p(r)^{(p(r)+1)\cdot p(r)} \cdot (p(r)+3)^{4p(r)} \leq (p(r)+3)^{(p(r)+6)p(r)}$$

This is asymptotically smaller than 2^{2^r}, the number of distinct Boolean functions over r variables. □

4.4. Modeling Graph Reachability with $DQBF^*$

In order to demonstrate how modeling can make use of the expressive power of dependency quantifiers and to further motivate our subsequent investigations of $DQBF^*$, we present in this section a $DQBF^*$ modeling pattern for reusing space by a multi-player game approach. With this technique, we develop a new $DQBF^*$ encoding of the bounded reachability problem for directed graphs and compare it to existing QBF^* representations [Pap94, DHK05].

4.4.1. Modeling Pattern: Saving Space with Multi-Player Games

Sections 1.1 and 2.6 have already shown that universal quantifiers can abbreviate multiple instantiations of (sub-)formulas. Let us now consider a more complex QBF^* example in which the subformulas that we wish to abbreviate also contain their own existentially quantified variables:

$$\begin{aligned}\Phi = \ & \big(\exists y_{1,1} \ldots \exists y_{1,l}\ \psi(z_{1,1}, \ldots, z_{1,k}, y_{1,1} \ldots, y_{1,l})\big) \\ \wedge\ & \big(\exists y_{2,1} \ldots \exists y_{2,l}\ \psi(z_{2,1}, \ldots, z_{2,k}, y_{2,1}, \ldots, y_{2,l})\big) \\ \wedge\ & \ldots \\ \wedge\ & \big(\exists y_{m,1} \ldots \exists y_{m,l}\ \psi(z_{m,1}, \ldots, z_{m,k}, y_{m,1}, \ldots, y_{m,l})\big)\end{aligned}$$

Here, the subformula ψ repeats m times, each time with a different set of free variables as leading parameters, followed by some unique existential variables. It should be clear that a prenex formula can easily be obtained by moving all quantifiers to the front. For a more compact QBF^* encoding, we now introduce

additional universally quantified auxiliary variables. We have the choice to declare them either in the scope of the existentials, or to put them in front. In the first case, this leads to

$$\exists y_{1,1}...\exists y_{m,l}\forall x_1...\forall x_{k+l}$$
$$\left(\bigvee_{i=1}^{m}\left(\bigwedge_{j=1}^{k}(x_j \leftrightarrow z_{i,j}) \wedge \bigwedge_{j=1}^{l}(x_{k+j} \leftrightarrow y_{i,j})\right)\right) \rightarrow \psi(x_1,...,x_{k+l})$$

where we first choose all the existentials and then abbreviate all arguments of ψ by the auxiliary variables. In the second case, we obtain the following formula:

$$\forall x_1...\forall x_k\exists y_1...\exists y_l \left(\bigvee_{i=1}^{m}\bigwedge_{j=1}^{k}(x_j \leftrightarrow z_{i,j})\right) \rightarrow \psi(x_1,...,x_k,y_1,...,y_l)$$

Now, the universals abbreviate only the first k arguments, and the values of the existential variables $y_1,...,y_l$ are overwritten for each of the m instantiations of ψ. It is clear that this requires less variables, and thus less space, than the first encoding. Still, both encodings are equivalent to the original formula, because in this example, the existentials in one subformula are independent from the existentials in another subformula. The second encoding is obviously not possible if we want to relate the existentials in one instance of ψ to the existentials in another instance, e.g. to enforce that the last l arguments are given different truth values for all instantiations of ψ. Is it possible to encode such relationships without saving all the existentials as in the first encoding? We will now see that *DQBF** makes this possible. In [PRA01], the behavior of a nondeterministic Turing machine over exponential time is encoded into a *DQBF** formula by tracking the machine's head movement and position, current and previous state, and current and previous symbol. It is important that those variables are properly updated, e.g., the encoding must enforce that $\mathbf{head}(\mathbf{t}+1) = \mathbf{head}(\mathbf{t}) + \mathbf{motion}(\mathbf{t})$ where $\mathbf{motion}(\mathbf{t}) \in \{-1,0,+1\}$. **head** and **t** are vectors of Boolean variables that store the corresponding information in a binary encoding, which is why we write them in bold. With the above modeling approach, it appears impossible to know the previous value of a variable. On the other hand, it is not feasible to store the values of the state variables for all time steps, because the machine may need exponential time and space. The solution of Peterson et al. to enforce a consistent computation without storing everything is to have two existential players. When both are given the same time by the universal player, they must both indicate the same head position, and if the given times differ by one, the difference between the head positions of the two players must be equal to the motion of the machine

at that time. That can be understood intuitively as carrying out the same computation twice, with an offset of one time tick, so that the leading computation thread can access values from the preceding step.

We suggest that this technique of having two independent existential players is not only useful for updating variables with subsequent values. It leads to a general pattern to avoid storing previous values in problems that require knowing the value of a variable y at time $\mathbf{t}_0$ and the value of the same variable at another time $\mathbf{t}_1$. If $\mathbf{t}_0$ and/or $\mathbf{t}_1$ are known ahead, it is easy to just store $y(\mathbf{t}_0)$ and/or $y(\mathbf{t}_1)$ in helper variables: $\forall\mathbf{t}\exists y\exists y_0\exists y_1\ ((\mathbf{t}=\mathbf{t}_0)\rightarrow(y=y_0))\wedge((\mathbf{t}=\mathbf{t}_1)\rightarrow(y=y_1))$. But if $\mathbf{t}_0$ and $\mathbf{t}_1$ are not known initially and/or may reoccur periodically, it is infeasible to store the values of y at all possibly affected times.

For example, assume that we wanted to check in the above Turing machine simulation whether there is a position on the tape on which the head is positioned more than once. That means we want to know whether there exist $\mathbf{t}_0$ and $\mathbf{t}_1$ with $\mathbf{t}_1>\mathbf{t}_0+1$ such that $\mathbf{head}(\mathbf{t}_0)=\mathbf{head}(\mathbf{t}_1)$ and $\mathbf{head}(\mathbf{t}_0+1)\neq\mathbf{head}(\mathbf{t}_0)$ (the head should really move to another position before returning). Since we do not know which pair of $\mathbf{t}_0$ and $\mathbf{t}_1$ we are looking for, it appears necessary to store all head positions from the beginning until we have found a matching pair. But with two existential players, we can assign each one a copy of the variable y that we are interested in (in the example, $y=\mathbf{head}$), say $y^{(1)}$ and $y^{(2)}$. They must be chosen independently at given times $\mathbf{t}^{(1)}$ and $\mathbf{t}^{(2)}$, which we can achieve with a *DQBF** prefix of the form $\forall\mathbf{t}^{(1)}\forall\mathbf{t}^{(2)}\exists y^{(1)}(\mathbf{t}^{(1)})\ \exists y^{(2)}(\mathbf{t}^{(2)})$. If $\mathbf{t}^{(1)}=\mathbf{t}^{(2)}$, both players must give the same values $y^{(1)}=y^{(2)}$. To work with values of y from different time points, we add conditions of the form

$$\Big((\mathbf{t}^{(1)}=\mathbf{t}_0)\wedge(\mathbf{t}^{(2)}=\mathbf{t}_1)\Big)\rightarrow\phi(y^{(1)},y^{(2)})$$

where ϕ is an expression that requires $y(\mathbf{t}_0)$ and $y(\mathbf{t}_1)$.

For the example of checking for duplicate head positions, we might get the following encoding:

$$\begin{aligned}
&\forall\mathbf{t}^{(1)}\forall\mathbf{t}^{(2)}\ \exists\mathbf{t}_0()\exists\mathbf{t}_1()\ \exists\mathbf{head}^{(1)}(\mathbf{t}^{(1)})\exists\mathbf{head}^{(2)}(\mathbf{t}^{(2)})\\
&(\mathbf{t}_1>\mathbf{t}_0+1)\wedge\mathit{simulate_machine}(\mathbf{t}^{(1)},\mathbf{t}^{(2)},\mathbf{head}^{(1)},\mathbf{head}^{(2)})\wedge\\
&\Big((\mathbf{t}^{(1)}=\mathbf{t}^{(2)})\rightarrow(\mathbf{head}^{(1)}=\mathbf{head}^{(2)})\Big)\wedge\\
&\Big((\mathbf{t}^{(1)}=\mathbf{t}_0)\wedge(\mathbf{t}^{(2)}=\mathbf{t}_1)\rightarrow(\mathbf{head}^{(1)}=\mathbf{head}^{(2)})\Big)\wedge\\
&\Big((\mathbf{t}^{(1)}=\mathbf{t}_0)\wedge(\mathbf{t}^{(2)}=\mathbf{t}_0+1)\rightarrow(\mathbf{head}^{(1)}\neq\mathbf{head}^{(2)})\Big)
\end{aligned}$$

This formula illustrates our introductory remarks that the most powerful feature of *DQBF** in comparison to *QBF** appears to be the ability to have variable dependencies which are independent of the formula structure: as far as the variable dependencies are concerned, $\mathbf{head}^{(1)}$ is completely unrelated to $\mathbf{head}^{(2)}$, yet both occur in common clauses of the formula and are indeed semantically related. It is unlikely that such patterns have a succinct *QBF** equivalent: since $\mathbf{head}^{(1)}$ and $\mathbf{head}^{(2)}$ occur in the same clauses, a non-prenex approach is not helpful here. And if we just kept the matrix of the formula above, but with a *QBF** prefix such as $\exists \mathbf{t}_0 \exists \mathbf{t}_1 \forall \mathbf{t}^{(1)} \exists \mathbf{head}^{(1)} \forall \mathbf{t}^{(2)} \exists \mathbf{head}^{(2)}$, the players might cheat. With this prefix, the second existential player may choose a move for the case $\mathbf{t}^{(2)} = \mathbf{t}_1$ and $\mathbf{t}^{(1)} = \mathbf{t}_0$ that differs from the one it claims to make when $\mathbf{t}^{(2)} = \mathbf{t}_1$, but $\mathbf{t}^{(1)} = \mathbf{t}_1$. That means the second player gives the first one false alibis that $\mathbf{head}^{(2)}(\mathbf{t}_1) = \mathbf{head}^{(1)}(\mathbf{t}_0)$ and $\mathbf{head}^{(2)}(\mathbf{t}_1) = \mathbf{head}^{(1)}(\mathbf{t}_1)$, but if $\mathbf{head}^{(2)}(\mathbf{t}_1)$ varies in the two statements, we have the undetected violation $\mathbf{head}^{(1)}(\mathbf{t}_0) \neq \mathbf{head}^{(1)}(\mathbf{t}_1)$.

In our example, we need to compare the value of **head** at three time points, $\mathbf{t}_0$, $\mathbf{t}_0 + 1$ and $\mathbf{t}_1$. We simulate this with two comparisons of two values each, but we might as well choose an encoding with three existential players:

$$
\begin{aligned}
&\forall \mathbf{t}^{(1)} \forall \mathbf{t}^{(2)} \mathbf{t}^{(3)}\ \exists \mathbf{t}_0() \exists \mathbf{t}_1()\ \exists \mathbf{head}^{(1)}(\mathbf{t}^{(1)}) \exists \mathbf{head}^{(2)}(\mathbf{t}^{(2)}) \exists \mathbf{head}^{(3)}(\mathbf{t}^{(3)}) \\
&(\mathbf{t}_1 > \mathbf{t}_0 + 1) \wedge \mathit{simulate\_machine}(\mathbf{t}^{(1)}, \mathbf{t}^{(2)}, \mathbf{head}^{(1)}, \mathbf{head}^{(2)}) \wedge \\
&\Big((\mathbf{t}^{(1)} = \mathbf{t}^{(2)} = \mathbf{t}^{(3)}) \rightarrow (\mathbf{head}^{(1)} = \mathbf{head}^{(2)} = \mathbf{head}^{(3)}) \Big) \wedge \\
&\Big((\mathbf{t}^{(1)} = \mathbf{t}_0) \wedge (\mathbf{t}^{(2)} = \mathbf{t}_1) \wedge (\mathbf{t}^{(3)} = \mathbf{t}_0 + 1) \rightarrow (\mathbf{head}^{(1)} = \mathbf{head}^{(2)} \neq \mathbf{head}^{(3)}) \Big)
\end{aligned}
$$

We cannot expect significant gains in expressiveness by using more than two existential players, because we can always simulate comparisons of multiple values by pairwise comparisons. But that might require additional helper variables as flags to remember intermediate comparison results. With additional players, we can do without them and, depending on the particular problem to be modeled, obtain more natural encodings.

4.4.2. The Bounded Reachability Problem

The bounded reachability problem for directed graphs is to decide whether a given graph has a path of bounded length from a start node *s* to a terminal node *t*.

Accordingly, the problem is also called *s-t-reachability*. The interpretation of the bound varies in literature: for a given integer k, the maximum path length is sometimes 2^k [Pap94] and sometimes k [DHK05]. We always use the former, and if necessary, we adapt existing QBF^* encodings to our definition for comparisons.

Definition 4.4.1. *(Bounded Reachability Problem for Directed Graphs)*
Let $G = (V,E)$ be a directed graph with a distinguished start node $s \in V$ and a terminal node $t \in V$, and let $k \geq 0$. Then the ***bounded reachability problem*** *is to decide whether G contains a path of length $\leq 2^k$ from s to t.*

In bounded model checking, currently the most prominent application, reachability of "bad" states that do not satisfy some given properties is usually checked with gradually increasing bound $k = 0, 1, 2, \ldots$. The process terminates whenever either a failure path to a bad state has been identified or a maximum bound has been reached. That maximum bound is sometimes a constant value, e.g. 10 or 20 for some experiments in [CBRZ01], when the graphs are huge and one can only expect to find short failure paths within reasonable computation time.

In order to find all failure paths, a sufficiently large maximum bound is the *recurrence diameter*, the length of the longest path without reoccurring states [BCCZ99]. But for large graphs, determining the recurrence diameter is a difficult problem itself (it can also be encoded into a (quantified) Boolean formula), and in the worst case, the recurrence diameter is close to the number of all nodes in the graph, which means reachability checking requires $k = O(n)$ to ensure completeness.

It is well known [Pap94, Sip05] that the reachability problem for directed graphs is solvable in $NSPACE(\log|V|)$ (or in deterministic $SPACE(\log^2|V|)$ by Savitch's Theorem [Sav70]), and clearly also in deterministic polynomial time, e.g. with a breadth-first search algorithm. This complexity classification assumes that the graph G is given explicitly, e.g. by an adjacency matrix. Such a representation may be very space-consuming for large graphs, but that is not reflected in the above space requirements, because the graph is part of the input and is therefore not counted as allocated space. For huge graphs, it might even be infeasible to provide an explicit representation. Notice that a graph whose vertices correspond to the state space of n binary variables has 2^n nodes, which leads to a state space explosion for growing n.

4.4.3. Existing Propositional and *QBF** Encodings

In applications like bounded model checking, graphs are typically extremely large, but also highly structured. With a more succinct encoding of graphs, it is possible to take advantage of this structuredness in order to obtain a much more compact representation of huge graphs that cannot be stored explicitly. Such symbolic encodings allow handling much larger problem instances, at the expense of an increase in complexity even for quite simple graph problems [GW84].

In Chapter 1, we have briefly introduced SAT-based bounded model checking, which currently appears to be one of the most promising symbolic approaches. Here, the nodes are considered as vectors of Boolean variables and the edges are implicitly given through a graph transition relation δ which is represented as a propositional formula. The reachability check then consists of determining the satisfiability of formulas like the following:

$$\phi := (\mathbf{s} = \mathbf{v}_0) \wedge (\mathbf{t} = \mathbf{v}_{2^k}) \bigwedge_{i=0}^{2^k-1} \delta(\mathbf{v}_i, \mathbf{v}_{i+1})$$

To illustrate that $\mathbf{s}$, $\mathbf{t}$ and $\mathbf{v}_i$ are actually bit vectors which store the vertices of the graph in a binary encoding, we write them in bold.

With this kind of encoding, we can easily extend *s-t*-reachability to the more general reachability problem with multiple start nodes $S \subseteq V$ and a set $T \subseteq V$ of terminal nodes where a path from some start node $s \in S$ to some terminal $t \in T$ is searched. The idea is to represent sets of nodes by characteristic functions:

$$\psi := S(\mathbf{v}_o) \wedge T(\mathbf{v}_{2^k}) \bigwedge_{i=0}^{2^k-1} \delta(\mathbf{v}_i, \mathbf{v}_{i+1})$$

where $S(\mathbf{v}_0) = 1$ if and only if $\mathbf{v}_0$ represents one of the start nodes, and analogously for T.

We can observe that there are 2^k copies of the transition relation δ, which can be quite space-consuming when we consider that δ represents the whole structure of the graph. We have already seen in Sections 1.2 and 2.6 that *QBF** encodings can eliminate multiple instantiations of a subformula for different arguments, so

the formula can be written in QBF^* with only one instance of δ [DHK05]:

$$\Psi(\mathbf{v}_0, \mathbf{v}_{2^k}) := \exists \mathbf{v}_1 ... \exists \mathbf{v}_{2^{k-1}} \forall \mathbf{u} \forall \mathbf{w}\, S(\mathbf{v}_0) \wedge T(\mathbf{v}_{2^k}) \wedge \left(\left(\bigvee_{i=0}^{2^k-1} ((\mathbf{u} = \mathbf{v}_i) \wedge (\mathbf{w} = \mathbf{v}_{i+1})) \right) \rightarrow \delta(\mathbf{u}, \mathbf{w}) \right)$$

Notice that Ψ has been formulated with free variables $\mathbf{v}_0$ and $\mathbf{v}_{2^k}$. This provides a more natural notation for recursive encodings like the next one, but if we are only interested in the mere existence of a path, Ψ can obviously be written as a closed QBF where $\mathbf{v}_0$ and $\mathbf{v}_{2^k}$ are also existentially quantified.

For simplicity, we assume that the number of nodes in the graph is a power of two. In the following discussion, we always let $|V| = 2^n$, which means each node in the graph needs n bits to be encoded. Then the formula Ψ above requires $O(n \cdot 2^k)$ quantified Boolean variables for given n and k. For the maximum bound $k = n$, we need exponentially many variables: $O(n \cdot 2^n)$. On the other hand, Ψ has a simple $\exists^* \forall^*$ prefix. It is possible to reduce the number of variables at the cost of a more complex prefix by applying iterative squaring [Pap94, DHK05] where paths of length 2^k can be checked in only k steps:

$$\begin{aligned}
\Phi(\mathbf{s}, \mathbf{t}) &:= S(\mathbf{s}) \wedge T(\mathbf{t}) \wedge \varphi_{2^k}(\mathbf{s}, \mathbf{t}) \\
\varphi_{2^k}(\mathbf{a}, \mathbf{b}) &:= \exists \mathbf{z} \forall \mathbf{u} \forall \mathbf{w} \\
&\quad ((\mathbf{u} = \mathbf{a}) \wedge (\mathbf{w} = \mathbf{z}) \vee (\mathbf{u} = \mathbf{z}) \wedge (\mathbf{w} = \mathbf{b})) \rightarrow \varphi_{2^{k-1}}(\mathbf{u}, \mathbf{w}) \\
\varphi_0(\mathbf{a}, \mathbf{b}) &:= (\mathbf{a} = \mathbf{b}) \vee \delta(\mathbf{a}, \mathbf{b})
\end{aligned}$$

This encoding requires only $O(n \cdot k)$ quantified variables for a graph with 2^n nodes and bound 2^k, but the prefix now has $O(k)$ quantifier alternations. For the maximum bound $k = n$, that means $O(n^2)$ variables in $O(n)$ quantifier blocks. We now demonstrate that $DQBF^*$ can encode the problem with even less variables, only $O(n+k) = O(n)$, while at the same time having a simple $\forall^* \exists^*$ prefix.

4.4.4. Developing a $DQBF^*$ Reachability Encoding

The QBF^* encodings that we have presented can be understood as two-player games. An existential player presents a series of vertices, and a universal player is checking them to make sure that they form a valid path according to the graph

transition relation. This checking can either occur in the end, after all vertices have been presented, as in encoding Ψ, or it can happen gradually, as in the formula Φ. In contrast to the *QBF** approaches, we now attempt to store only one single pair of vertices at a time. That means we need a step counter $\mathbf{c}$, which is conveniently encoded into a vector of universally quantified variables, and existentially quantified vectors $\mathbf{u}$ and $\mathbf{v}$ that are chosen depending on the value of $\mathbf{c}$. This approach requires that all consistency checks must be made in a step-wise fashion.

An important correctness criterion for reachability is to have a continuous path: after going from some vertex $\mathbf{u}$ to a vertex $\mathbf{v}$, we cannot continue in the next step with a transition from $\mathbf{w}$ to $\mathbf{z}$ if $\mathbf{v} \neq \mathbf{w}$. Apparently, continuity cannot be checked when only one pair of vertices is known. The multi-player modeling pattern from the beginning of this section provides a solution to the problem of reusing variables while still requiring access to earlier values. Accordingly, we have one universal and two non-cooperating existential players. Each existential player i is presented its own universally quantified step counter $\mathbf{c}^{(i)}$. Depending on the value of the counter, it then chooses vertices $\mathbf{u}^{(i)}$ and $\mathbf{v}^{(i)}$ to indicate a move from $\mathbf{u}^{(i)}$ to $\mathbf{v}^{(i)}$ at the $\mathbf{c}^{(i)}$-th step ($i = 1,2$). We obtain a prefix of the following form:

$$\forall \mathbf{c}^{(1)} \forall \mathbf{c}^{(2)}\ \exists \mathbf{u}^{(1)}, \mathbf{v}^{(1)}(\mathbf{c}^{(1)})\ \exists \mathbf{u}^{(2)}, \mathbf{v}^{(2)}(\mathbf{c}^{(2)})$$

As in the previous section, we assume that $|V| = 2^n$, which means the $\mathbf{u}^{(i)}$ and $\mathbf{v}^{(i)}$ need to be vectors of n Boolean variables each. For a path length of 2^k, the $\mathbf{c}^{(i)}$ need k bits each. To maintain similarity with the *QBF** representations from above, we go for an encoding with free variables $\mathbf{s}$ and $\mathbf{t}$. In the formula matrix, we must then verify that $\mathbf{s}$ and $\mathbf{t}$ are indeed start and terminal nodes, respectively, and we must check that the existential players start in $\mathbf{s}$ and finish in $\mathbf{t}$. Finally, we need the conditions that the players behave identically when the counters are the same and make continuous transitions when the counters differ by one.

Then our complete *DQBF** encoding is as follows:

$$\begin{aligned}
\Gamma(\mathbf{s},\mathbf{t}) :=\ & \forall \mathbf{c}^{(1)} \forall \mathbf{c}^{(2)}\ \exists \mathbf{u}^{(1)}, \mathbf{v}^{(1)}(\mathbf{c}^{(1)})\ \exists \mathbf{u}^{(2)}, \mathbf{v}^{(2)}(\mathbf{c}^{(2)}) \\
& S(\mathbf{s}) \wedge T(\mathbf{t}) \wedge \delta(\mathbf{u}^{(1)}, \mathbf{v}^{(1)}) \wedge \\
& ((\mathbf{c}^{(1)} = 0) \rightarrow (\mathbf{u}^{(1)} = \mathbf{s})) \wedge \\
& ((\mathbf{c}^{(1)} = 2^k - 1) \rightarrow (\mathbf{v}^{(1)} = \mathbf{t})) \wedge \\
& ((\mathbf{c}^{(2)} = \mathbf{c}^{(1)}) \rightarrow (\mathbf{u}^{(1)} = \mathbf{u}^{(2)}) \wedge (\mathbf{v}^{(1)} = \mathbf{v}^{(2)})) \wedge \\
& ((\mathbf{c}^{(2)} = \mathbf{c}^{(1)} + 1) \rightarrow (\mathbf{v}^{(1)} = \mathbf{u}^{(2)}))
\end{aligned}$$

The correctness of the encoding is easy to verify: if existential player 1 goes from some vertex v_i to v_j at the r-th step, existential player 2 must continue at the $(r+1)$-th step from v_j to, say, v_l. Both players must behave identically, thus it follows that player 1 also continues from v_j to v_l at step $(r+1)$.

4.4.5. Comparisons

Just like the QBF^* encodings Φ and Ψ, we can get along with only one copy of the transition relation δ and the indicator functions S and T. Besides this unavoidable embedded description of the graph, our encoding Γ is more concise, as the comparison in Table 4.1 shows: the matrix of Γ contains only a constant number of tests for equality, whereas Φ needs $O(k)$ and Ψ even $O(2^k)$ equality checks. More importantly, we need less quantified variables: Γ has $O(k)$ universal and $O(n)$ existential variables, while the QBF^* formula Φ has $O(n \cdot k)$ existential and also $O(n \cdot k)$ universal variables. Recall that the latter encoding has the strong disadvantage of having a complex prefix with $O(k)$ quantifier alternations. Ψ requires as much as $O(n \cdot 2^k)$ existential and $O(n)$ universal variables. For the maximum bound $k = n$, we have a total of $O(n)$ quantified variables in the $DQBF^*$ encoding versus the QBF^* formulas with $O(n^2)$ (in an unbounded number of quantifier blocks) or even $O(2^n)$ variables.

Table 4.1.: Comparison of bounded reachability encodings

	$\Gamma \in DQBF^*$	$\Phi \in QBF^*$	$\Psi \in QBF^*$
Equality Checks	$O(1)$	$O(k)$	$O(2^k)$
Existential Variables	$O(n)$	$O(n \cdot k)$	$O(n \cdot 2^k)$
Universal Variables	$O(k)$	$O(n \cdot k)$	$O(n)$
Quantifier Alternations	$O(1)$	$O(k)$	$O(1)$

The compactness of the new encoding Γ comes with the price of the potentially higher complexity of $DQBF^*$. However, it will be shown in Section 4.6.2 that instances of Γ for classes of graphs with maximum bound $k = O(\log|\delta|)$ are in a Σ_2^P-complete subclass of $DQBF^*$. With the graph having $|V| = 2^n$ nodes, we can well assume that δ has size at least $\Omega(\log|V|) = \Omega(n)$ when we only consider non-trivial graphs. Then Γ is solvable in polynomial space when the maximum

bound is in $O(\log n)$. That is sufficient for applications in which the recurrence diameter is at most polynomial in n, or if exponentially long paths are simply not feasible for very large graphs. As mentioned earlier, there are hard problems for which bounded model checking appears only practical with small constant maximum bounds [CBRZ01].

The above condition $k = O(\log|\delta|)$ is also satisfied when we have maximum bounds with $k = O(n)$, which guarantees completeness regardless of the diameter, and graphs with $|\delta| = \Omega(|V|) = \Omega(2^n)$. Such graphs must certainly exist, because there are $2^{2^{2n}}$ possible adjacency matrices over 2^n vertices, but at most $(p(n)+3)^{4p(n)}$ different propositional formulas of length $p(n)$ [KBL99]. The latter is asymptotically smaller than $2^{2^{2n}}$, so there are graphs that cannot be encoded with a poly-length transition relation. Unfortunately, the exponential growth of δ makes such problems barely manageable for sufficiently large values of n, which means the main advantage of symbolic methods over techniques with explicit enumeration of vertices is lost. However, there might still be applications with graphs that are highly structured for the most part, with some small loosely structured components that have no succinct encoding. Then δ would still grow exponentially, but with much smaller constant factors than explicit graph representations.

Since there are no *DQBF** solvers available yet, it is hard to predict for real problem instances whether the conciseness and simple structure of the *DQBF** encoding will win over more voluminous *QBF** formulas with a possibly easier satisfiability problem. But we can observe that both *QBF** encodings above have structural features that are known for having negative effects on the actual performance of solvers: Ψ has exponentially many variables in the worst case, whereas Φ has an unbounded number of quantifier alternations. The former is obviously unpleasant for solvers, but the latter is also problematic, because the variable selection of the solvers is usually limited to individual quantifier blocks and is therefore drastically restricted when the variables are distributed over a large number of different blocks. Our *DQBF** encoding, on the other hand, is not only shorter with asymptotically fewer equality checks, but has the benefits of a clear and simple prefix with a lower number of variables.

4.5. Universal Quantifier Expansion for *DQBF**

The expansion of universal quantifiers is a key technique in our work on *QBF** formulas. We now demonstrate that universal expansion can also be applied in an analogous way to *DQBF** formulas.

4.5.1. Expansion Procedure and Correctness

As in *QBF**, the idea of universal expansion is that a universal quantifier $\forall x\, \phi(x)$ is just an abbreviation for $\phi(0) \wedge \phi(1)$. We can expand it by making two copies of the original matrix, one for the universally quantified variable being false, and one for that variable being true. Existential variables which depend on that universal variable need to be duplicated as well. Consider an example:

$$\forall x_1 \forall x_2 \forall x_3 \exists y_1(x_1,x_2) \exists y_2(x_1,x_3) \exists y_3(x_3)\, \phi(x_1,x_2,x_3,y_1,y_2,y_3,\mathbf{z})$$

We want to expand the universal x_1, on which the existentials y_1 and y_2 depend. We must therefore introduce two separate instances $y_{1,(0)}$ and $y_{1,(1)}$ of the original variable y_1, where $y_{1,(0)}$ is used in the copy of the matrix for $x_1 = 0$, and $y_{1,(1)}$ for $x_1 = 1$. Analogously, y_2 is duplicated into $y_{2,(0)}$ and $y_{2,(1)}$. We obtain the following expanded formula:

$$\begin{aligned}&\forall x_2 \forall x_3 \exists y_{1,(0)}(x_2) \exists y_{1,(1)}(x_2) \exists y_{2,(0)}(x_3) \exists y_{2,(1)}(x_3) \exists y_3(x_3)\\ &\phi(0,x_2,x_3,y_{1,(0)},y_{2,(0)},y_3,\mathbf{z}) \wedge \phi(1,x_2,x_3,y_{1,(1)},y_{2,(1)},y_3,\mathbf{z})\end{aligned}$$

The dependency lists immediately indicate which existentials must be duplicated when a universal variable is expanded. In that respect, universal expansion for *DQBF** formulas is in fact clearer and more natural than its *QBF** equivalent. On the other hand, the correctness of the expansion procedure is easy to see in *QBF**, because the semantics definition of *QBF** explicitly handles universal quantifiers with the condition that $\mathfrak{I}(\forall x \Phi') = 1 \Leftrightarrow \mathfrak{I}(\Phi[x/0]) = \mathfrak{I}(\Phi[x/1]) = 1$ for any interpretation $\mathfrak{I}$ [KBL99]. According to Section 4.3, the *DQBF** semantics, however, is defined in a more implicit way by using model functions. A little more effort is therefore needed to prove that the universal quantifiers in *DQBF** still behave according to our intuitive understanding and that the expansion procedure outlined above is indeed correct.

Theorem 4.5.1. *(Correctness of Universal Expansion for DQBF*)*
Let Φ be a DQBF formula in which we want to expand the innermost universal quantifier $\forall x_n$. Without loss of generality, we assume that the existential variables are arranged in two blocks, depending on whether they are dominated by x_n or not:*

$$\begin{aligned}\Phi(\mathbf{z}) \;=\; & \forall x_1 ... \forall x_n \exists y_1(\mathbf{x}_{\mathbf{d}_1}) ... \exists y_k(\mathbf{x}_{\mathbf{d}_k}) \exists y_{k+1}(\mathbf{x}_{\mathbf{d}_{k+1}}, x_n) ... \exists y_m(\mathbf{x}_{\mathbf{d}_m}, x_n) \\ & \phi(x_1, ..., x_n, y_1, ..., y_m, \mathbf{z})\end{aligned}$$

with $x_n \notin \mathbf{x}_{\mathbf{d}_i}$ for all $1 \leq i \leq m$.

Then $\Phi(\mathbf{z}) \approx \Phi'(\mathbf{z})$ where

$$\begin{aligned}\Phi'(\mathbf{z}) \;=\; & \forall x_1 ... \forall x_{n-1} \exists y_1(\mathbf{x}_{\mathbf{d}_1}) ... \exists y_k(\mathbf{x}_{\mathbf{d}_k}) \\ & \exists y_{k+1,(0)}, y_{k+1,(1)}(\mathbf{x}_{\mathbf{d}_{k+1}}) ... \exists y_{m,(0)}, y_{m,(1)}(\mathbf{x}_{\mathbf{d}_m}) \\ & \phi(x_1, ..., x_{n-1}, 0, y_1, ..., y_k, y_{k+1,(0)}, ..., y_{m,(0)}, \mathbf{z}) \wedge \\ & \phi(x_1, ..., x_{n-1}, 1, y_1, ..., y_k, y_{k+1,(1)}, ..., y_{m,(1)}, \mathbf{z})\end{aligned}$$

is the formula obtained from expanding $\forall x_n$.

Proof:
In order to show $\Phi(\mathbf{z}) \approx \Phi'(\mathbf{z})$, we must prove that $\Phi(t(\mathbf{z})) = 1 \Leftrightarrow \Phi'(t(\mathbf{z})) = 1$ for any truth assignment $t(\mathbf{z}) := (t(z_1), ..., t(z_r)) \in \{0,1\}^r$ to the free variables $\mathbf{z} = (z_1, ..., z_r)$. For any given $t(\mathbf{z})$, we can consider $\Phi(t(\mathbf{z}))$ and $\Phi'(t(\mathbf{z}))$ as closed *DQBF* formulas which are true if and only if they have a satisfiability model.

From left to right: let $M = (f_{y_1}, ..., f_{y_m})$ be a satisfiability model for $\Phi(t(\mathbf{z}))$, and let $\phi(x_1, ..., x_n, f_{y_1}, ..., f_{y_m}, t(\mathbf{z}))$ denote the formula that we obtain when all occurrences of y_i in ϕ are replaced with the formula $f_{y_i}(x_{d_{i,1}}, ..., x_{d_{i,n_i}})$ for each $i = 1, ..., m$, and when the free variables $\mathbf{z}$ are assigned the truth values $t(\mathbf{z})$. Define $G_{(0)} := (g_{y_1}, ..., g_{y_k}, g_{y_{k+1,(0)}}, ..., g_{y_{m,(0)}})$ with $g_{y_i} := f_{y_i}$ for $i = 1, ..., k$ and $g_{y_{i,(0)}}(x_{d_{i,1}}, ..., x_{d_{i,n_i}}) := f_{y_i}(x_{d_{i,1}}, ..., x_{d_{i,n_i}}, 0)$ for $i = k+1, ..., m$. Then

$$\forall x_1 ... \forall x_n \; \phi(x_1, ..., x_n, f_{y_1}, ..., f_{y_m}, t(\mathbf{z})) = 1$$

implies

$$\forall x_1 ... \forall x_{n-1} \; \phi(x_1, ..., x_{n-1}, 0, g_{y_1}, ..., g_{y_k}, g_{y_{k+1,(0)}}, ..., g_{y_{m,(0)}}, t(\mathbf{z})) = 1 \; .$$

Similarly, we let $G_{(1)} := (g_{y_1}, ..., g_{y_k}, g_{y_{k+1,(1)}}, ..., g_{y_{m,(1)}})$ with model functions $g_{y_{i,(1)}}(x_{d_{i,1}}, ..., x_{d_{i,n_i}}) := f_{y_i}(x_{d_{i,1}}, ..., x_{d_{i,n_i}}, 1)$ for $i = k+1, ..., m$, such that

$$\forall x_1 ... \forall x_{n-1}\ \phi(x_1, ..., x_{n-1}, 1, g_{y_1}, ..., g_{y_k}, g_{y_{k+1,(1)}}, ..., g_{y_{m,(1)}}, t(\mathbf{z})) = 1\ .$$

It follows that $G = (g_{y_1}, ..., g_{y_k}, g_{y_{k+1,(0)}}, g_{y_{k+1,(1)}}, ..., g_{y_{m,(0)}}, g_{y_{m,(1)}})$ is a satisfiability model for $\Phi'(t(\mathbf{z}))$.

From right to left: let $G = (g_{y_1}, ..., g_{y_k}, g_{y_{k+1,(0)}}, g_{y_{k+1,(1)}}, ..., g_{y_{m,(0)}}, g_{y_{m,(1)}})$ be a satisfiability model for $\Phi'(t(\mathbf{z}))$. We now construct a model $M = (f_{y_1}, ..., f_{y_m})$ that satisfies $\Phi(t(\mathbf{z}))$. For $i = 1, ..., k$, we let $f_{y_i} := g_{y_i}$, and for $i = k+1, ..., m$, we define

$$f_{y_i}(x_{d_{i,1}}, ..., x_{d_{i,n_i}}, x_n) := (x_n \vee g_{y_{i,(0)}}(x_{d_{i,1}}, ..., x_{d_{i,n_i}})) \wedge (\neg x_n \vee g_{y_{i,(1)}}(x_{d_{i,1}}, ..., x_{d_{i,n_i}}))$$

such that $f_{y_i}[x_n/0] := f_{y_i}(x_{d_{i,1}}, ..., x_{d_{i,n_i}}, 0) \approx g_{y_{i,(0)}}(x_{d_{i,1}}, ..., x_{d_{i,n_i}})$, and thus:

$$\begin{aligned} & \forall x_1 ... \forall x_{n-1}\ \phi(x_1, ..., x_{n-1}, 0, f_{y_1}, ..., f_{y_k}, f_{y_{k+1}}[x_n/0], ..., f_{y_m}[x_n/0], t(\mathbf{z})) \\ \approx\ & \forall x_1 ... \forall x_{n-1}\ \phi(x_1, ..., x_{n-1}, 0, g_{y_1}, ..., g_{y_k}, g_{y_{k+1,(0)}}, ..., g_{y_{m,(0)}}, t(\mathbf{z})) \end{aligned}$$

The latter is true, because G is a satisfiability model for $\Phi'(t(\mathbf{z}))$. The case $x_n = 1$ works analogously, and it follows that

$$\forall x_1 ... \forall x_{n-1} \forall x_n\ \phi(x_1, ..., x_{n-1}, x_n, f_{y_1}, ..., f_{y_m}, t(\mathbf{z})) = 1$$

which confirms that M is a satisfiability model for $\Phi(t(\mathbf{z}))$. □

4.5.2. Iterated Expansion and *DQBF** to *QBF** Transformation

We can apply the universal expansion procedure from Theorem 4.5.1 iteratively and eliminate multiple universal quantifiers. While the number of existential variables grows when doing so, their dependency lists are getting shorter, because when a universal x_i is expanded, it is of course removed from all dependencies. If the process continues, we ultimately obtain a formula with only existential quantifiers, all of them having empty dependencies. Then the existential variables have simple 0/1 satisfiability models, so that the formula can also be

considered as a $\exists BF^*$ formula. That means we have just found a transformation from $DQBF^*$ to $\exists BF^* \subseteq QBF^*$.

In Section 4.3.3, we have already shown that $DQBF^*$ formulas can be converted into QBF^* by replacing existential variables with the corresponding equivalence model functions, but with universal expansion, we do not need to know the equivalence model.

Theorem 4.5.2. *For every $DQBF^*$ formula*

$$\Phi(\mathbf{z}) = \forall x_1 ... \forall x_n \exists y_1(x_{d_{1,1}}, ..., x_{d_{1,n_1}}) ... \exists y_m(x_{d_{m,1}}, ..., x_{d_{m,n_m}})\, \phi(\mathbf{x}, \mathbf{y}, \mathbf{z})$$

with universal variables $\mathbf{x} = x_1, ..., x_n$, existential variables $\mathbf{y} = y_1, ..., y_m$ and free variables $\mathbf{z}$, there exists an equivalent $\exists BF^*$ formula $\Phi_{\exists BF}$ that can be obtained by universal quantifier expansion as follows:

$$\begin{aligned}\Phi_{\exists BF}(\mathbf{z}) \;:=\; & \exists y_{1,(0,...,0)} \exists y_{1,(0,...,0,1)} ... \exists y_{1,(1,...,1,0)} \exists y_{1,(1,...,1)} \\ & \vdots \\ & \exists y_{m,(0,...,0)} \exists y_{m,(0,...,0,1)} ... \exists y_{m,(1,...,1,0)} \exists y_{m,(1,...,1)} \\ & \bigwedge_{t(\mathbf{x}) \in \{0,1\}^n} \phi(t(\mathbf{x}), y_{1,(t(x_{d_{1,1}}),...,t(x_{d_{1,n_1}}))}, ..., y_{m,(t(x_{d_{m,1}}),...,t(x_{d_{m,n_m}}))}, \mathbf{z})\end{aligned}$$

Proof:
The equivalence of Φ and $\Phi_{\exists BF}$ immediately follows from the fact that this transformation is an iterated application of individual expansion steps, which have been shown to be equivalence-preserving in Theorem 4.5.1. □

Here is an example: the formula

$$\Phi(\mathbf{z}) = \forall x_1 \forall x_2 \forall x_3 \exists y_1(x_1, x_2) \exists y_2(x_2, x_3)\, \phi(x_1, x_2, x_3, y_1, y_2, \mathbf{z})$$

is expanded to

$$\begin{aligned}\Phi_{\exists BF}(\mathbf{z}) \;=\; & \exists y_{1,(0,0)} \exists y_{1,(0,1)} \exists y_{1,(1,0)} \exists y_{1,(1,1)} \exists y_{2,(0,0)} \exists y_{2,(0,1)} \exists y_{2,(1,0)} \exists y_{2,(1,1)} \\ & \phi(0,0,0, y_{1,(0,0)}, y_{2,(0,0)}, \mathbf{z}) \wedge \phi(0,0,1, y_{1,(0,0)}, y_{2,(0,1)}, \mathbf{z}) \\ \wedge\; & \phi(0,1,0, y_{1,(0,1)}, y_{2,(1,0)}, \mathbf{z}) \wedge \phi(0,1,1, y_{1,(0,1)}, y_{2,(1,1)}, \mathbf{z}) \\ \wedge\; & \phi(1,0,0, y_{1,(1,0)}, y_{2,(0,0)}, \mathbf{z}) \wedge \phi(1,0,1, y_{1,(1,0)}, y_{2,(0,1)}, \mathbf{z}) \\ \wedge\; & \phi(1,1,0, y_{1,(1,1)}, y_{2,(1,0)}, \mathbf{z}) \wedge \phi(1,1,1, y_{1,(1,1)}, y_{2,(1,1)}, \mathbf{z})\end{aligned}$$

If we just want to obtain a *QBF** formula, but not necessarily $\exists$*BF**, we do not need to expand all universals. When we only expand x_1 in the formula $\Phi(\mathbf{z})$ from above, we get

$$\begin{aligned}\Phi'(\mathbf{z}) \;=\; & \forall x_2 \forall x_3 \exists y_{1,(0)}(x_2) \exists y_{1,(1)}(x_2) \exists y_2(x_2, x_3) \\ & \phi(0, x_2, x_3, y_{1,(0)}, y_2, \mathbf{z}) \wedge \phi(1, x_2, x_3, y_{1,(1)}, y_2, \mathbf{z})\end{aligned}$$

which can already be written in prenex *QBF** as

$$\Psi(\mathbf{z}) = \forall x_2 \exists y_{1,(0)} \exists y_{1,(1)} \forall x_3 \exists y_2 \, \phi(0, x_2, x_3, y_{1,(0)}, y_2, \mathbf{z}) \wedge \phi(1, x_2, x_3, y_{1,(1)}, y_2, \mathbf{z}) \,.$$

More details about the selection of universals to expand when transforming *DQBF** formulas into *QBF** will be provided in Section 4.6.1. But even with sophisticated selection strategies, it is clear that this expansion may lead to an exponential blowup of the input formula, which is not surprising if we assume that *PSPACE* $\neq$ *NEXPTIME*. An example for a formula with exponential growth is our *DQBF** encoding of bounded reachability from Section 4.4.4 with the following prefix:

$$\forall \mathbf{c}^{(1)} \forall \mathbf{c}^{(2)} \; \exists \mathbf{u}^{(1)}, \mathbf{v}^{(1)}(\mathbf{c}^{(1)}) \; \exists \mathbf{u}^{(2)}, \mathbf{v}^{(2)}(\mathbf{c}^{(2)})$$

In order to convert this *DQBF** formula Γ into a *QBF** formula, we must expand at least one of the universally quantified vectors $\mathbf{c}^{(1)}$ or $\mathbf{c}^{(2)}$, both of which may have $O(|\Gamma|)$ many universal variables in the worst case.

We will, however, show in Sections 4.6.1 and 4.7 that there are non-trivial subclasses of *DQBF** for which the above expansion remains polynomial.

4.6. Structure of the Dependencies

In this and the next section, we attempt to avoid the *NEXPTIME*-completeness of *DQBF** by considering interesting *DQBF** subclasses that have lower complexity. In *QBF**, the most well-known subclasses are defined through constraints on the clause structure or through restrictions on the structure of the quantifier prefix. The latter typically means that the sequence of quantifier blocks is restricted to a particular prefix type. However, *DQBF** formulas can always be written with a $\forall^*\exists^*$ prefix, so this approach is not useful in the *DQBF** case. We have

already seen that the information contained in quantifier alternations in QBF^* is represented in the dependency lists $x_{d_{i,1}},...,x_{d_{i,n_i}}$ of dependency quantified existentials y_i. Our idea is therefore to obtain $DQBF^*$ subclasses by imposing restrictions on the structure of the dependencies.

4.6.1. Orderable Dependencies

We can identify the class of (prenex) QBF^* formulas with the class of $DQBF^*$ formulas $\Phi(\mathbf{z}) = \forall x_1...\forall x_n \exists y_1(\mathbf{x_{d_1}})...\exists y_m(\mathbf{x_{d_m}})\ \phi(\mathbf{x},\mathbf{y},\mathbf{z})$ that satisfy the condition ($\mathbf{x_{d_j}} \subseteq \mathbf{x_{d_k}}$ or $\mathbf{x_{d_j}} \supseteq \mathbf{x_{d_k}}$) for all $1 \leq j,k \leq m$, because that implies a total ordering $\mathbf{x_{d_{i_1}}} \subseteq ... \subseteq \mathbf{x_{d_{i_m}}}$ which can be represented by a linear prefix with non-dependency quantifiers $\forall \mathbf{x_{d_{i_1}}} \exists y_1 \forall (\mathbf{x_{d_{i_2}}} \setminus \mathbf{x_{d_{i_1}}}) \exists y_2 .... \forall (\mathbf{x_{d_{i_m}}} \setminus \mathbf{x_{d_{i_{m-1}}}}) \exists y_m$, where $\forall \mathbf{x_{d_i}} := \forall x_{d_{i,1}}...\forall x_{d_{i,n_i}}$.

It is easy to see that classes of QBF^* formulas with a particular prefix type, that means formulas with a restricted number of quantifier blocks in the prefix, correspond to $DQBF^*$ formulas that satisfy the ordering condition above and the additional restriction that the number of different dependencies is bounded.

With the help of the universal quantifier expansion method from Section 4.5, we can eliminate all universals which violate the total ordering. If that affects only logarithmically many universals, they can be expanded in polynomial time and thus in polynomial space. Then we can weaken the ordering requirements as follows:

Definition 4.6.1. *(Polynomially Orderable Dependencies $D_{po}QBF^*$)*
With $D_{po}QBF^*$, we denote the class of dependency quantified Boolean formulas with ***polynomially orderable dependencies****. That is, formulas*

$$\Phi(\mathbf{z}) = \forall x_1...\forall x_n \exists y_1(\mathbf{x_{d_1}})...\exists y_m(\mathbf{x_{d_m}})\ \phi(\mathbf{x},\mathbf{y},\mathbf{z})$$

with

$$\left| \bigcup_{1 \leq j < k \leq m} (\mathbf{x_{d_{s_j}}} \setminus \mathbf{x_{d_{s_k}}}) \right| \in O(\log |\Phi|)$$

for some ordering $S = (s_1,...,s_m) \in \{1,...,m\}^m$ on the dependencies.

Theorem 4.6.2. *The satisfiability problem for $D_{po}QBF^*$ is PSPACE-complete.*

Proof:
Given $\Phi \in D_{po}QBF^*$, let Φ' be the formula obtained from Φ by expanding all universal quantifiers in $X := \bigcup_{1 \le j < k \le m} (\mathbf{x}_{\mathbf{d}_{s_j}} \setminus \mathbf{x}_{\mathbf{d}_{s_k}})$. Then $|\Phi'| \le 2^{|X|} \le p(|\Phi|)$ for some polynomial p, and Φ' can be computed in polynomial time, because each expansion step can be performed in time at most $O(|\Phi'|^2)$. The remaining dependencies $\mathbf{x}'_{\mathbf{d}_1}, ..., \mathbf{x}'_{\mathbf{d}_{m'}}$ in Φ' can be ordered $\mathbf{x}'_{\mathbf{d}_{i_1}} \subseteq ... \subseteq \mathbf{x}'_{\mathbf{d}_{i_{m'}}}$. As discussed above, Φ' can thus be rewritten as an equivalent QBF^* formula, which can be solved in polynomial space.
The *PSPACE*-hardness is easily seen from the fact that we can write in polynomial time any QBF^* formula Ψ as an equivalent $DQBF^*$ formula Ψ' in which the dependencies $\mathbf{x}_{\mathbf{d}_i}$ for each existential quantifier $\exists y_i$ are exactly those universals whose quantifiers occur earlier in the prefix of Ψ. Then $|\Psi'| \le |\Psi|^2$ and $\mathbf{x}_{\mathbf{d}_j} \subseteq \mathbf{x}_{\mathbf{d}_k}$ for all $j < k$, which is just a special case of the definition above. □

An application of this discussion might be a satisfiability solver or proof system which externally accepts $D_{po}QBF^*$ formulas, but internally preprocesses them into QBF^* and feeds them to an encapsulated QBF^* engine. That would give users the benefit of a clearer notation with explicit variable dependencies, which removes the need for non-prenex formulas and allows for more natural modeling of problems, while the system itself could be based on existing QBF^* techniques. We will later see that the translation from dependency quantifiers to ordinary quantifiers is also possible in polynomial time for Horn formulas.

4.6.2. Dependencies of Limited Size

Another obvious restriction on the dependencies is to limit their cardinalities as in the following definition:

Definition 4.6.3. *(Logarithmic Dependencies $D_{log}QBF^*$)*
*With $D_{log}QBF^*$, we denote the class of dependency quantified Boolean formulas with **logarithmic dependencies**. That is, formulas*

$$\Phi(\mathbf{z}) = \forall x_1 ... \forall x_n \exists y_1 (x_{d_{1,1}}, ..., x_{d_{1,n_1}}) ... \exists y_m (x_{d_{m,1}}, ..., x_{d_{m,n_m}})\ \phi(\mathbf{x}, \mathbf{y}, \mathbf{z})$$

with $n_1, ..., n_m \in O(log\ |\Phi|)$, such that each existential variable depends on at most logarithmically many universals.

Theorem 4.6.4. *The satisfiability problem for $D_{log}QBF^*$ is Σ_2^P-complete.*

Proof:
We first show that $D_{log}QBF^*$-SAT is in Σ_2^P. Recall that the complexity class $\Sigma_2^P = NP^{NP}$ contains all problems which can be decided non-deterministically in polynomial time with the help of an *NP* oracle or, equivalently, a *coNP* oracle. We begin with guessing a satisfying truth assignment $\tau(\mathbf{z})$ to the free variables. Then we can replace in polynomial time all occurrences of free variables z_i with the corresponding truth value $\tau(z_i)$, so that we can continue with the closed formula $\Phi' = \Phi(\tau(\mathbf{z}))$. Φ' is satisfiable if and only if it has a satisfiability model $M = (f_{y_1}, ..., f_{y_m})$. Each f_{y_i} can be represented as a truth table with $O(2^{log\ |\Phi|}) = O(|\Phi|)$ rows and $O(log\ |\Phi|)$ columns or, equivalently, as a propositional formula of length $O(|\Phi| \cdot log\ |\Phi|)$. Hence, the whole satisfiability model can be guessed in time and space $O(m \cdot |\Phi| \cdot log\ |\Phi|) = O(|\Phi|^3)$.
We can then replace in polynomial time each occurrence of an existential y_i with the corresponding f_{y_i} represented as propositional formula. The resulting formula $\Phi'[\mathbf{y}/M]$ is an ordinary QBF^* formula with only universal quantifiers and has size $O(|\Phi|^3)$. Its satisfiability can be checked with a *coNP*-oracle.

The Σ_2^P-hardness of $D_{log}QBF^*$-SAT is easy to prove with a reduction from the satisfiability problem for $QBF_{2,\exists}$, the class of quantified Boolean formulas with a $\exists^*\forall^*$ prefix. A $QBF_{2,\exists}$ formula

$$\Psi = \exists y_1 ... \exists y_m \forall x_1 ... \forall x_n\ \psi(x_1, ..., x_n, y_1, ..., y_m)$$

is satisfiable if and only if the corresponding $D_{log}QBF^*$ formula

$$\Phi = \forall x_1 ... \forall x_n \exists y_1() ... \exists y_m()\ \phi(x_1, ..., x_n, y_1, ..., y_m)$$

is satisfiable[1]. □

A natural special case of $D_{log}QBF^*$ is what we call D_kQBF^* formulas, in which the dependencies are bounded by a maximum cardinality k.

Definition 4.6.5. *(Bounded Dependencies D_kQBF^*)*
For fixed $k \geq 0$, let D_kQBF^* denote the class of dependency quantified Boolean formulas with ***bounded dependencies****. That is, formulas*

$$\Phi = \forall x_1 ... \forall x_n \exists y_1(x_{d_{1,1}}, ..., x_{d_{1,n_1}}) ... \exists y_m(x_{d_{m,1}}, ..., x_{d_{m,n_m}})\ \phi(\mathbf{x}, \mathbf{y})$$

with $n_1, ..., n_m \leq k$, such that an existential variable can depend on at most k universal variables.

[1] We could avoid the empty dependencies by adding a dummy variable x_0 as follows: $\forall x_0 \forall x_1 ... \forall x_n \exists y_1(x_0) ... \exists y_m(x_0)\ \phi(x_1, ..., x_n, y_1, ..., y_m)$

It is clear that the Σ_2^P-hardness in the preceding proof still holds for D_kQBF^*, since the reduced formula has existential variables which depend on zero universals (or at most one, if the remark from the footnote is applied).

Corollary 4.6.6. *The satisfiability problem for D_kQBF^* is Σ_2^P-complete.*

Problems where the dependencies are restricted to logarithmic or constant length may occur quite naturally within the general pattern for $DQBF^*$ modeling that we have presented in Section 4.4.1. In this game-based approach with a universal player and multiple existential players, the universal player provides counter values encoded into binary vectors $\mathbf{c}^{(i)}$, and the i-th existential player has to state its actions in round $\mathbf{c}^{(i)}$. The matrix of the formula is satisfiable if and only if the actions of the existential players are consistent with the rules of the game and the existential players reach a final (winning) state. It is easy to see that such formulas are in $D_{log}QBF^*$ if the range of the counter values is asymptotically at most logarithmic in the size of the consistency and termination checks that make up the formula matrix.

For a specific example, consider our encoding Γ of bounded reachability from Section 4.4.4. Γ has a prefix of the form

$$\forall \mathbf{c}^{(1)} \forall \mathbf{c}^{(2)} \, \exists \mathbf{u}^{(1)}, \mathbf{v}^{(1)}(\mathbf{c}^{(1)}) \, \exists \mathbf{u}^{(2)}, \mathbf{v}^{(2)}(\mathbf{c}^{(2)})$$

with $\mathbf{c}^{(i)} = (c_1^{(i)}, ..., c_k^{(i)})$ being vectors of k Boolean variables that can encode step counters from 0 to $2^k - 1$, and $\mathbf{u}^{(i)}, \mathbf{v}^{(i)} \in \{0,1\}^n$ being vectors of n variables to encode up to 2^n graph nodes. That means each existential variable depends on $O(k)$ universals. The formula matrix of our reachability encoding grows asymptotically at least as fast as the transition relation δ (assuming we have non-trivial graphs with $|\delta| \in \Omega(\log|V|) = \Omega(n)$). It follows that $\Gamma \in D_{log}QBF^*$ for classes of instances with $k \in O(\log|\delta|)$.

An interesting observation is that we also have $\Gamma \in D_{po}QBF^*$ if $k \in O(\log|\delta|)$. That means we could apply universal expansion to transform such formulas Γ into equivalent QBF^* formulas of polynomial length. The same observation applies to all formulas following the above multi-player game approach when the counter values have logarithmic bounds and there is only a constant number of existential players. If, however, the number of existential players is not constant, but depends on n, the formulas are still in $D_{log}QBF^*$, but no longer in $D_{po}QBF^*$, and universal expansion will cause exponential growth. Moreover, in the case

of the bounded reachability encoding Γ, it is probably not a good idea anyway to perform universal expansion in this way, because it would involve duplicating the transition relation δ many times, which was exactly what we wanted to prevent with a quantified encoding in the first place.

To conclude this section, we would like to point out that the *DQBF** encoding of bounded reachability has a remarkably simple structure: there are two vectors of universally quantified counter variables $\mathbf{c}^{(1)}$ and $\mathbf{c}^{(2)}$, and all existential variables depend on either $\mathbf{c}^{(1)}$ or $\mathbf{c}^{(2)}$. That means such formulas have only two different dependency lists (in multiple instantiations), and the two are even disjoint. Nevertheless, this is sufficient to reason about paths with up to 2^n steps, which is exponentially larger than the formula size for succinctly representable graphs. The expressiveness of this simple construction hardly leaves any possibilities to achieve lower complexity by making any other restrictions on the structure of the dependencies besides the ones that have been suggested here.

4.7. Dependency Quantified Horn Formulas

We now consider formulas with Horn clauses, one of the most interesting restrictions on the structure of clauses, in contrast to the restrictions on the prefix structure from the preceding section. Formally, we define the class *DQHORN** to contain all *DQBF** formulas in conjunctive normal form whose clauses have at most one positive literal. Besides quantification with dependencies, this is analogous to the class *QHORN** that we have studied thoroughly in Chapter 3, so we can investigate how the addition of dependency quantifiers affects the expressive power and the complexity of the satisfiability problem.

4.7.1. Satisfiability Models for *DQHORN* Formulas

As in Chapter 3, we begin our investigations with closed *DQHORN* formulas and the corresponding satisfiability models. Again, we denote by B_n^i the bit vector of length n where only the i-th element is zero. Moreover, $Z_{\leq 1}(n)$, $Z_{=1}(n)$, and $Z_{\geq 1}(n)$, respectively, are again tuples of length n with at most one zero, exactly one zero, and at least one zero, respectively.

In Theorem 3.3.5, we have been able to show that the satisfiability model of a *QHORN* formula can be reduced to a small core, namely a $Z_{\leq 1}$-partial satisfiability model, which covers only cases where at most one of the universally quantified variables is zero. That result could be considered as a generalization of the property of propositional Horn formulas that the intersection of two satisfying truth assignments is also a satisfying truth assignment. Can we further lift this to dependency quantified Horn formulas? It appears that the closure under intersection is such a fundamental property of Horn formulas that it can indeed be applied to *DQHORN* as well. The following approach of completing partial models into total satisfiability models resembles the *QHORN* case, but we have chosen a more elementary proof to make it clearer that the dependencies are respected.

Definition 4.7.1. *($R_\forall$-partial Satisfiability Model for DQBF)*
For $\Phi = \forall x_1 ... \forall x_n \exists y_1(\mathbf{x_{d_1}}) ... \exists y_m(\mathbf{x_{d_m}})\ \phi(\mathbf{x},\mathbf{y})$, let $M = (f_{y_1}, ..., f_{y_m})$ map each existential variable y_i to a propositional formula f_{y_i} over the universal variables $\mathbf{x_{d_i}} = x_{d_{i,1}}, ..., x_{d_{i,n_i}}$ on which y_i depends. Furthermore, let $R_\forall(n)$ be a relation on the set of possible truth assignments to n universals. Then M is a ***$R_\forall$-partial satisfiability model*** *for Φ if the formula $\phi[\mathbf{y}/M]$ is true for all $\mathbf{x} \in R_\forall(n)$.*

Theorem 4.7.2. *Let $\Phi = Q\phi(\mathbf{x},\mathbf{y}) \in DQHORN$ be a dependency quantified Horn formula with a $Z_{\leq 1}$-partial satisfiability model $M = (f_{y_1}, .., f_{y_m})$. Then its total completion $M^t = (f^t_{y_1}, ..., f^t_{y_m})$ with*

$$\begin{aligned} f^t_{y_i}(x_{d_{i,1}}, ..., x_{d_{i,n_i}}) := \quad & (x_{d_{i,1}} \vee f_{y_i}(0,1,1,...,1)) \\ \wedge\ & (x_{d_{i,2}} \vee f_{y_i}(1,0,1,...,1)) \\ \wedge\ & ... \\ \wedge\ & (x_{d_{i,n_i}} \vee f_{y_i}(1,1,...,1,0)) \\ \wedge\ & f_{y_i}(1,...,1) \end{aligned}$$

is a satisfiability model for Φ.

Proof:
We need to show that $\phi[\mathbf{y}/M^t]$ is true for all possible truth value assignments $t(\mathbf{x}) := (t(x_1), ..., t(x_n)) \in \{0,1\}^n$ to the universal variables.
Since $f^t_{y_j}(1,...,1) = f_{y_j}(1,...,1)$, we only need to consider $t(\mathbf{x}) \in Z_{\geq 1}$.

The proof is by induction on the number of zeros in $t(\mathbf{x})$. The induction base is the case $t(\mathbf{x}) \in Z_{=1}$. Let $t(\mathbf{x}) = B^i_n$ be an assignment to the universals where

$t(x_i) = 0$. In order to prove that every clause in $\phi[\mathbf{y}/M^t]$ is true for $t(\mathbf{x})$, we make a case distinction on the structure of Horn clauses. Any clause C belongs to one of the following cases:

1. $C = y_j \vee \bigvee_{l \in L_y} \neg y_l \vee \bigvee_{l \in L_x} \neg x_l$ (positive existential literal y_j):
   We only consider $i \notin L_x$, because $C[\mathbf{y}/M^t]$ is trivially true if $i \in L_x$.
   If $f_{y_j}(t(x_{d_{j,1}}), ..., t(x_{d_{j,n_j}})) = 1$ and $f_{y_j}(1, ..., 1) = 1$ then
   $f^t_{y_j}(t(x_{d_{j,1}}), ..., t(x_{d_{j,n_j}})) = f_{y_j}(t(x_{d_{j,1}}), ..., t(x_{d_{j,n_j}})) \wedge f_{y_j}(1, ..., 1) = 1$.
   Otherwise, if $f_{y_j}(\tau) = 0$ for $\tau = (t(x_{d_{j,1}}), ..., t(x_{d_{j,n_j}}))$ and/or $\tau = (1, ..., 1)$, we have $f_{y_r}(\tau) = 0$ for some $r \in L_y$, as M is a $Z_{\leq 1}$-partial satisfiability model. This implies $f^t_{y_r}(t(x_{d_{r,1}}), ..., t(x_{d_{r,n_r}})) = 0$, which makes $C[\mathbf{y}/M^t]$ true.

2. $C = x_j \vee \bigvee_{l \in L_x} \neg x_l \vee \bigvee_{l \in L_y} \neg y_l$ (positive universal literal x_j):
   The only interesting case is $i = j$. M being a $Z_{\leq 1}$-partial satisfiability model implies $f_{y_r}(t(x_{d_{r,1}}), ..., t(x_{d_{r,n_r}})) = 0$ for some $r \in L_y$. It follows that $f^t_{y_r}(t(x_{d_{r,1}}), ..., t(x_{d_{r,n_r}})) = 0$.

3. $C = \bigvee_{l \in L_x} \neg x_l \vee \bigvee_{l \in L_y} \neg y_l$ (no positive literal):
   We only need to discuss $i \notin L_x$. Then this case is analogous to 2.

For the induction step, we consider an assignment where $k > 1$ universals are false. Let $t(x_{i_1}) = 0$, ..., $t(x_{i_k}) = 0$ and $t(x_s) = 1$ for $s \neq i_1, ..., i_k$. In order to show that $\phi[\mathbf{y}/M^t]$ is true for $t(\mathbf{x})$, we can use the induction hypothesis and assume that $\phi[\mathbf{y}/M^t]$ is true for $t_1(\mathbf{x}) = B^{i_k}_n$ as well as for $t_{k-1}(\mathbf{x})$ with $t_{k-1}(x_1) = 0$, ..., $t_{k-1}(x_{i_{k-1}}) = 0$ and $t_{k-1}(x_s) = 1$ for $s \neq i_1, ..., i_{k-1}$. That means the case with k zeros x_{i_1},..., x_{i_k} is reduced to the case where only x_{i_k} is zero and the case where x_{i_1}, ..., $x_{i_{k-1}}$ are zero.

Then the definition of $f^t_{y_j}$ implies:

$$f^t_{y_j}(t(x_{d_{j,1}}), ..., t(x_{d_{j,n_j}})) = f^t_{y_j}(t_1(x_{d_{j,1}}), ..., t_1(x_{d_{j,n_j}})) \wedge f^t_{y_j}(t_{k-1}(x_{d_{j,1}}), ..., t_{k-1}(x_{d_{j,n_j}}))$$

Again, we make a case distinction. It is actually very similar to the one from the induction base:

1. $C = y_j \vee \bigvee_{l \in L_y} \neg y_l \vee \bigvee_{l \in L_x} \neg x_l$ (positive existential literal y_j):
   We assume $i_1, ..., i_k \notin L_x$, because otherwise, $C[\mathbf{y}/M^t]$ is trivially true.
   If $f^t_{y_j}(t_1(x_{d_{j,1}}), ..., t_1(x_{d_{j,n_j}})) = 1$ and $f^t_{y_j}(t_{k-1}(x_{d_{j,1}}), ..., t_{k-1}(x_{d_{j,n_j}})) = 1$, we have $f^t_{y_j}(t(x_{d_{j,1}}), ..., t(x_{d_{j,n_j}})) = 1$.

Otherwise, without loss of generality, let $f^t_{y_j}(t_1(x_{d_{j,1}}),...,t_1(x_{d_{j,n_j}})) = 0$. Then the induction hypothesis implies that $f^t_{y_r}(t_1(x_{d_{r,1}}),...,t_1(x_{d_{r,n_r}})) = 0$ for some $r \in L_y$, and we get $f^t_{y_r}(t(x_{d_{r,1}}),...,t(x_{d_{r,n_r}})) = 0$.

2. $C = x_j \vee \bigvee_{l\in L_x} \neg x_l \vee \bigvee_{l\in L_y} \neg y_l$ (positive universal literal x_j):
   The only interesting case to discuss is $j \in \{i_1,...,i_k\}$. Without loss of generality, we assume $j = i_k$. Then it follows from the induction hypothesis that $f^t_{y_r}(t_1(x_{d_{r,1}}),...,t_1(x_{d_{r,n_r}})) = 0$ for some $r \in L_y$, and therefore $f^t_{y_r}(t(x_{d_{r,1}}),...,t(x_{d_{r,n_r}})) = 0$.
3. $C = \bigvee_{l\in L_x} \neg x_l \vee \bigvee_{l\in L_y} \neg y_l$ (no positive literal):
   We only need to discuss $i_1,...,i_k \notin L_x$. Then this case is analogous to 2.

□

It follows that *DQHORN* formulas have satisfiability models with functions of the form $f_{y_i}(x_{d_{i,1}},...,x_{d_{i,n_i}}) = \bigwedge_{j\in J} x_{d_{i,j}}$ (or the constants $f_{y_i} = 0$ resp. $f_{y_i} = 1$), just the same K_2 structure that is characteristic for the class *QHORN*. This result leads to the surprising observation that Horn formulas with dependency quantifiers are not substantially more expressive than with ordinary quantifiers. Accordingly, we are going to show in the next section that the former can be transformed into the latter without significant blowup.

4.7.2. Transformation from $DQHORN^*$ to $\exists HORN^*$

Earlier in this chapter, we have described how universal quantifier expansion can eliminate dependencies and convert $DQBF^*$ formulas into equivalent QBF^* formulas, with obvious exponential blowup for formulas with arbitrary variable dependencies. But from Section 3.6.2, we already know that the expansion of universal quantifiers is feasible for ordinary $QHORN^*$ formulas, because the behavior of the existential quantifiers is completely determined by the cases where at most one universal is false. With the result from the previous section on closed *DQHORN* formulas, which we can also apply to $DQHORN^*$ by considering fixed assignments to the free variables, we can now prove that it is also feasible to transform $DQHORN^*$ formulas into $\exists HORN^*$ by universal expansion.

Theorem 4.7.3. *(DQHORN* to ∃HORN* Transformation)*
Any DQHORN formula*

$$\Phi(\mathbf{z}) = \forall x_1 ... \forall x_n \exists y_1(x_{d_{1,1}}, ..., x_{d_{1,n_1}}) ... \exists y_m(x_{d_{m,1}}, ..., x_{d_{m,n_m}})\, \phi(\mathbf{x}, \mathbf{y}, \mathbf{z})$$

can be transformed into an ∃HORN formula*

$$\begin{aligned}\Phi_{\exists HORN}(\mathbf{z}) \;:=\; & \exists y_{1,(0,1,...,1)} \exists y_{1,(1,0,1,...,1)} ... \exists y_{1,(1,...,1,0)} \exists y_{1,(1,...,1)} \\ & \vdots \\ & \exists y_{m,(0,1,...,1)} \exists y_{m,(1,0,1,...,1)} ... \exists y_{m,(1,...,1,0)} \exists y_{m,(1,...,1)} \\ & \bigwedge_{t(\mathbf{x}) \in Z_{\leq 1}(n)} \phi(t(\mathbf{x}), y_{1,(t(x_{d_{1,1}}),...,t(x_{d_{1,n_1}}))}, ..., y_{m,(t(x_{d_{m,1}}),...,t(x_{d_{m,n_m}}))}, \mathbf{z})\end{aligned}$$

such that $\Phi(\mathbf{z}) \approx \Phi_{\exists HORN}(\mathbf{z})$.

Example:

$$\Phi(\mathbf{z}) = \forall x_1 \forall x_2 \forall x_3 \exists y_1(x_1, x_2) \exists y_2(x_2, x_3)\, \phi(x_1, x_2, x_3, y_1, y_2, \mathbf{z})$$

with matrix $\phi \in HORN$ is transformed into

$$\begin{aligned}\Phi_{\exists HORN}(\mathbf{z}) \;=\; & \exists y_{1,(0,1)} \exists y_{1,(1,0)} \exists y_{1,(1,1)} \exists y_{2,(0,1)} \exists y_{2,(1,0)} \exists y_{2,(1,1)} \\ & \phi(0,1,1, y_{1,(0,1)}, y_{2,(1,1)}, \mathbf{z}) \wedge \phi(1,0,1, y_{1,(1,0)}, y_{2,(0,1)}, \mathbf{z}) \\ \wedge\; & \phi(1,1,0, y_{1,(1,1)}, y_{2,(1,0)}, \mathbf{z}) \wedge \phi(1,1,1, y_{1,(1,1)}, y_{2,(1,1)}, \mathbf{z})\end{aligned}$$

Proof:
The implication $\Phi(\mathbf{z}) \models \Phi_{\exists HORN}(\mathbf{z})$ is obvious, as the clauses in $\Phi_{\exists HORN}$ are just a subset of the clauses in the general *DQBF** to *∃BF** expansion $\Phi_{\exists BF}$, which has been shown in Theorem 4.5.2 to be equivalent to Φ.

The implication $\Phi_{\exists HORN}(\mathbf{z}) \models \Phi(\mathbf{z})$ is more interesting. We first consider the case $\Phi \in DQHORN$ without free variables and prove that the satisfiability of $\Phi_{\exists HORN}$ implies that Φ has a $Z_{\leq 1}$-partial satisfiability model. Let t be a satisfying truth assignment to the existentials in $\Phi_{\exists HORN}$. From t, we construct a $Z_{\leq 1}$-partial satisfiability model for Φ by assembling the truth assignments to

the individual copies $y_{i,(x_{d_{i,1}},\ldots,x_{d_{i,n_i}})}$ of an existential variable y_i into a common model function:

$$\begin{aligned} f_{y_i}(x_{d_{i,1}},\ldots,x_{d_{i,n_i}}) = \quad & (\neg x_{d_{i,1}} \wedge x_{d_{i,2}} \wedge \ldots \wedge x_{d_{i,n_i}} \rightarrow t(y_{i,(0,1,\ldots,1)})) \\ \wedge\ & (x_{d_{i,1}} \wedge \neg x_{d_{i,2}} \wedge x_{d_{i,3}} \wedge \ldots \wedge x_{d_{i,n_i}} \rightarrow t(y_{i,(1,0,1,\ldots,1)})) \\ \wedge\ & \ldots \\ \wedge\ & (x_{d_{i,1}} \wedge \ldots \wedge x_{d_{i,n_i-1}} \wedge \neg x_{d_{i,n_i}} \rightarrow t(y_{i,(1,\ldots,1,0)})) \\ \wedge\ & (x_{d_{i,1}} \wedge \ldots \wedge x_{d_{i,n_i}} \rightarrow t(y_{i,(1,\ldots,1)})) \end{aligned}$$

Now, the f_{y_i} constitute a $Z_{\leq 1}$-partial satisfiability model for Φ, because for all $\mathbf{x} = (x_1,\ldots,x_n)$ with $\mathbf{x} \in Z_{\leq 1}$, we have $f_{y_i}(x_{d_{i,1}},\ldots,x_{d_{i,n_i}}) = t(y_{i,(x_{d_{i,1}},\ldots,x_{d_{i,n_i}})})$, and $\phi(x_1,\ldots,x_n,t(y_{1,(x_{d_{1,1}},\ldots,x_{d_{1,n_1}})}),\ldots,t(y_{m,(x_{d_{m,1}},\ldots,x_{d_{m,n_m}})})) = 1$ due to the satisfiability of $\Phi_{\exists HORN}$.

In the case that $\Phi \in DQHORN^*$ has free variables, let $\mathbf{z}^*$ an arbitrary assignment to the free variables such that $\Phi_{\exists HORN}(\mathbf{z}^*)$ is satisfiable. With the free variables fixed, we can treat both $\Phi_{\exists HORN}(\mathbf{z}^*)$ and $\Phi(\mathbf{z}^*)$ as closed formulas. Then we have just shown in the preceding paragraph of this proof that the satisfiability of $\Phi_{\exists HORN}(\mathbf{z}^*)$ implies that $\Phi(\mathbf{z}^*)$ has a $Z_{\leq 1}$-partial satisfiability model. Then it follows from Theorem 4.7.2 that $\Phi(\mathbf{z}^*)$ has a satisfiability model and is therefore satisfiable. □

We immediately obtain the following corollary:

Corollary 4.7.4. *For any $DQHORN^*$ formula Φ, there exists an equivalent $\exists HORN^*$ formula without dependency quantifiers whose length is bounded by $|\forall| \cdot |\Phi|$, where $|\forall|$ is the number of universal quantifiers in Φ, and $|\Phi|$ is the length of Φ.*

The slow growth of the resulting formula and its simple prefix with only existential non-dependency quantifiers make this transformation suitable for determining the satisfiability of a formula $\Phi \in DQHORN^*$ with the following algorithm:

1. Transform Φ into $\Phi_{\exists HORN}$ according to Theorem 4.7.3. This requires time $O(|\forall| \cdot |\Phi|)$ and produces a formula of length $|\Phi_{\exists HORN}| = O(|\forall| \cdot |\Phi|)$.
2. Determine the satisfiability of $\phi_{\exists HORN}$, the matrix of $\Phi_{\exists HORN}$. This propositional Horn formula can be solved in time $O(|\phi_{\exists HORN}|) = O(|\forall| \cdot |\Phi|)$.

In total, the algorithm requires time $O(|\forall| \cdot |\Phi|)$, which means *DQHORN* satisfiability is not only tractable, but appears to have the same complexity as the best known algorithms for solving $QHORN^*$.

Theorem 4.7.5. *Let $\Phi \in DQHORN^*$ be a dependency quantified Horn formula with free variables. Then the satisfiability of Φ can be determined within time $O(|\forall| \cdot |\Phi|)$, where $|\forall|$ is the number of universal quantifiers in Φ, and $|\Phi|$ is the length of Φ.*

In the proof of Theorem 4.7.3, it is not required that the free variables have the Horn property. We can therefore extend our result to $DQCNF^*$ formulas in which only the quantified variables alone need to be a Horn formula, which means every clause may have an arbitrary number of positive free literals and at most one positive quantified literal. We call these formulas $DQHORN^b$, in analogy to Section 3.8. In combination with earlier results that $\exists HORN^b$ and $QHORN^b$ satisfiability are *NP*-complete and that both classes are poly-time equivalent, we obtain the following theorem:

Theorem 4.7.6. *The satisfiability problem for the formula class $DQHORN^b$ is NP-complete, and $DQHORN^b =_{poly-time} QHORN^b =_{poly-time} \exists HORN^b$.*

Our results show that explicit variable dependencies can be added to quantified Horn formulas without increase in complexity. On the other hand, dependency quantified Horn formulas are not significantly more concise, only by a quadratic factor. But we obtain for free the inherent benefits of dependency quantification, such as a clearer notation that removes the need for non-prenex formulas and allows for more natural modeling of problems.

It is important to point out that these results apply only to quantified $HORN^b$ formulas. For arbitrary quantified Boolean formulas, it is generally assumed that more powerful quantification is exponentially more concise, that means $DQBF^* >_{poly-length} QBF^* >_{poly-length} \exists BF^*$, unless the corresponding complexity classes coincide. Proving this chain of increasing expressiveness without complexity-theoretic assumptions appears, however, to be a very challenging problem.

4.8. Summary

Partially ordered quantification with explicit variable dependencies is a powerful addition to the language of quantified Boolean formulas. It allows more succinct and more natural formula representations without the need for non-prenex formulas. In particular, the ability to have variable dependencies that are independent from the formula structure allows for new compact modeling approaches. By extending the well-known two-player game encodings from QBF^* with additional independent existentially quantified players, $DQBF^*$ encodings can better reuse quantified variables. We have demonstrated this with a new $DQBF^*$ representation of the well-known bounded reachability problem for directed graphs which is fundamental to bounded model checking.

The benefits of dependency quantification can be enjoyed without the higher complexity of general $DQBF^*$ satisfiability when suitable subclasses are considered. We show that interesting non-trivial subclasses are obtained through restrictions on the clause structure or the structure of the dependencies in the prefix. The latter includes formulas with dependencies of bounded or at most logarithmic size (D_kQBF^* or $D_{log}QBF^*$) which are Σ_2^P-complete to solve, as well as formulas with dependencies that can be ordered within polynomial space, denoted $D_{po}QBF^*$, a *PSPACE*-complete generalization of QBF^*.

Our most interesting result on $DQBF^*$ subclasses is that quantified Horn formulas do not appear to become more complex when dependency quantifiers are added. The closure under intersection of satisfying truth assignments proves to be a fundamental property of Horn clauses, and our framework of partial satisfiability models seems to capture this quite well, so that we could smoothly lift our main results from Chapter 3 to $DQHORN^*$ and $DQHORN^b$.

We have also shown that universal quantifier expansion works just like in QBF^*, in fact even slightly more elegant, because it is explicitly indicated which existentials are dominated by a universal variable. As we have demonstrated, that allows translating $DQBF^*$ into QBF^*, so that a solver or proof system could be built which uses $DQBF^*$ as a specification language, but solves such formulas by preprocessing them into QBF^* and using an encapsulated QBF^* solver. It is shown that the QBF^* translation requires only polynomial time for $DQHORN^*$, $DQHORN^b$ and $D_{po}QBF^*$ formulas.

Besides the conversion of $DQBF^*$ into QBF^*, universal expansion might also be helpful in building a full $DQBF^*$ solver, because after all, expansion is one of

the most successful solving techniques for QBF^*. Adopting other approaches, e.g. DPLL-style backtracking, to $DQBF^*$ appears to be problematic because of the necessity to store all the model functions associated with the existentially quantified variables. However, one of the best available *QBF* solvers, sKizzo [Ben05a], is based on explicitly computing the model functions and storing them compactly as ROBDDs. We assume that this approach might also be successfully lifted to $DQBF^*$, so that it should only be a matter of time until we see expansion- and/or model-based $DQBF^*$ solvers being published. Our results on easier subclasses, in particular the tractability of $DQHORN^*$, might further benefit such solvers by allowing them to cut easier subproblems from a larger computation.

5. Bounded Universal Expansion

In this chapter, we present a new approach for preprocessing $QCNF^*$ formulas by expanding a selection of universally quantified variables with bounded expansion costs. We describe a suitable selection strategy which combines cost estimates with goal orientation by taking into account unit literals which might be obtained.

In order to reduce the overhead from duplicating existential variables which depend on the universal being expanded, we present a technique to compute compact sets of dependent existentials by taking into account the local connectivity of variables and their polarity. In the best case, this can reduce the number of dependent existentials by an arbitrarily large factor in comparison to existing approaches.

Furthermore, we investigate how Q-resolution can be integrated into our preprocessing. In particular, resolution is applied specifically on the dependent existentials to further reduce their number.

Experimental results on well-known problems from the QBFLIB formula collection demonstrate that our preprocessing is very effective on simplifying the prefixes and can significantly improve the performance of state-of-the-art *QBF* solvers.

This chapter extends preliminary results on preprocessing by universal expansion published in [BKB07].

5.1. Motivation

So far, one of the topics that we have particularly focused on is the expansion of universal quantifiers. Although repeated application can lead to exponential blowup in the worst case, universal expansion appears to be typically cheaper for *QCNF** formulas than the elimination of existential quantifiers by expansion, Q-resolution or symbolic skolemization. This imbalance between universal and existential quantification in clausal formulas has been very evident for the various classes of Horn formulas that we have considered. But besides such tractable special cases, the addition of universal quantifiers to existentially quantified formulas can still make solving the formulas much more difficult. From a theoretical viewpoint, it causes a climb to higher levels in the polynomial hierarchy. And from a practical viewpoint, a purely existentially quantified formula $\exists y_1 ... \exists y_m \; \phi(\mathbf{y}, \mathbf{z})$ can be solved by invoking a SAT solver on the propositional matrix ϕ. But the addition of a universal quantifier $\forall x$ generally requires two calls to the SAT solver. Similarly, a QDPLL-based solver typically needs to branch for both possible values of x. At the same time, the variable ordering imposed by the nesting of the quantifiers must be respected, which severely limits the effectiveness of look-ahead and variable selection heuristics. Not only search algorithms are affected in this way, but also the other approaches like symbolic skolemization, because the arity of the model functions increases when the value of an existential variable y_i depends on the value of an additional universal.

Our idea is therefore to specifically attack the universally quantified variables in *QCNF** formulas by universal expansion. While this operation is certainly exponential in nature for arbitrary *QCNF**, we are going to develop refinements that make it more manageable. In addition, we are now going to avoid the problem of exponential growth by not eliminating all universals in the formula. We suggest a preprocessing by bounded universal expansion which eliminates only a suitably selected subset of universals with bounded expansion costs. This is often sufficient to significantly simplify the prefixes of given formulas: a look at existing encodings, e.g. in the QBFLIB collection [GNT01], confirms that *QCNF** instances from various application domains typically have significantly less universal quantifiers than existentials, and for formulas with multiple alternations of quantifiers, the universal blocks usually tend to be rather short. In this situation, a relatively small number of expansions can be very effective.

5.2. Research Goals and Related Work

The main goal of this chapter is to develop in detail the concept of bounded universal expansion that was suggested in the previous paragraph, in order to turn it into a powerful preprocessing approach for *QCNF** formulas. The underlying idea, which we also want to verify experimentally with a suitable implementation, is to make it significantly easier for subsequently invoked solvers by first taking out those universals which are cheap to expand and/or particularly rewarding. The other universals remain as they are, in the hope that the solvers might be able to handle them better with a different approach.

To our knowledge, universal expansion has been used successfully in at least two *QBF** solvers: QUBOS by Ayari and Basin [AB02] and Quantor by Armin Biere [Bie05]. QUBOS is not restricted to *CNF* formulas and can switch to a dual mode where it only expands existentials by $\exists y\ \phi(y) \approx \phi(0) \vee \phi(1)$. This is easy to do without the *CNF* requirement, but we prefer to stay with *CNF*. On the one hand, our initial hypothesis about focusing on the universals only applies to clausal formulas, and on the other hand, we want to avoid a retransformation after preprocessing, because most solvers still require *CNF* as well. In addition, we are going to make use of the specific clause structure for further universal expansion refinements.

Unlike the bounded expansion that we suggest, both systems ultimately expand all universals (unless the formula collapses prematurely) and then solve the remaining purely existentially quantified formula with an ordinary SAT solver. While Quantor expands universals from the innermost universal quantifier block in a prenex formula, QUBOS uses quantifier shifting rules [ETW02, NW01] like *miniscoping* by the equivalence $\forall x\ (\Psi(x) \wedge \Gamma(x)) \approx (\forall x\ \Psi(x)) \wedge (\forall x\ \Gamma(x))$ to drive quantifiers inside the formula. QUBOS then expands quantifiers from the innermost scopes of the resulting non-prenex formula.

Our idea is to lift the restriction to innermost scopes that both solvers have, because we will see in the next section that universals which are particularly rewarding for expansion might also occur further outside in the prefix. While QUBOS and Quantor will eventually get to these universals, our bounded expansion might reach the bound before that happens. To make sure that all rewarding universals are considered, we extend universal expansion by allowing the selection of universals from the whole prefix. In order to take advantage of this larger

degree of freedom, we use a refined scheduling strategy which also takes into account possible expansion benefits, such as the generation of unit literals.

In addition to the bounding, we attempt to mitigate the exponential nature of universal expansion by reducing its actual costs. In Sections 3.6 and 4.5, the expansion of a universal quantifier has required us to duplicate all existentials which depend on that universal. On the one hand, we suggest to reduce the number of such dependent existentials by specifically applying Q-resolution on them before making the expansion, rather than using Q-resolution as an alternative to universal expansion, which is what Quantor does. On the other hand, we want to improve the dependency computation itself, which means we do not want to treat existentials as being dependent on a universal if they can in fact be chosen independently. In *DQBF**, we had all dependencies given explicitly, which was a big advantage. But for *QBF** formulas, the simple approach of duplicating all existentials whose quantifiers occur in a prefix block to the right of the universal quantifier being expanded is too coarse, especially when expanding non-innermost universals.

Determining whether an existential variable y_i depends on a certain universal x_j in a given formula is equivalent to asking whether the formula has a satisfiability or equivalence model in which the model function f_{y_i} does not depend on x_j. The problem can easily be solved in *PSPACE* if we simply expand x_j (following the procedure given in the upcoming Theorem 5.3.1 on Page 133) in two different ways, once with duplicating the existential y_i, and once without doing so, and then check the equivalence of the two resulting formulas in polynomial space. On the other hand, it is easy to see that this problem is at least as difficult as determining the satisfiability of a *QBF* formula: with an arbitrary *QBF* formula $\Phi = Qv_1 ... Qv_k\ \phi(v_1, ..., v_k)$, we can associate the formula $\Psi := \forall x \exists y Q v_1 ... Q v_k\ (x \vee y) \wedge (\neg x \vee \neg y \vee \phi(v_1, ..., v_k))$ with new variables x and y. Then Ψ has a satisfiability model with $f_y = 1$ if and only if Φ is true, which proves the *PSPACE*-hardness.

Faced with this high complexity, an important question is whether we can efficiently compute tight supersets of dependent existentials. [Bie05] suggests for a universal x_j from the innermost universal quantifier block to duplicate all innermost existentials which occur in a common clause with x_j. Then those existentials again propagate dependency to all other existentials from the innermost block that they share a clause with, and so on. In the end, the set of dependent existentials is given by the transitive closure of this local connectivity relation.

We extend the idea with three contributions:

1. We generalize it to work with universals from the whole prefix and prove the correctness of the resulting dependency concept with the help of quantifier shifting [ETW02, NW01].

2. We add the ability to split large universal scopes by miniscoping.

3. We make the important observation that in addition to variable connectivity in common clauses, the polarity of variables can also have a great influence on dependency. This allows us to prove the correctness of a stronger dependency concept that can achieve a reduction by an arbitrarily large factor in the best case.

[Ben05b] presents an alternative approach, but without variable polarity (Item 3) and thus with more redundancy in general. The idea of the approach is to build a tree-like structure by gradually merging scopes of existential quantifiers. We will later see how this is related to the computation of local connectivity. The trees are grown from the existentials to the universals, whereas our approach propagates dependency from universals to existentials. We argue that the latter direction is more suitable for our universal expansion approach. It is also unclear how quantifier trees could be combined with variable polarity to achieve a similar level of conciseness.

A result that is very similar to the combination of Items 1 and 3 (i.e. Definition 5.3.2 together with Theorems 5.3.3 and 5.6.1 below) has been found independently and simultaneously with our initial publication [BKB07] by Samer and Szeider in [SS07]. As explained later in Section 5.6, they essentially use a different notation where universals can also be called dependent on existentials.

Some of our ideas have already been picked up by recent papers: [LB08a] presents a generalization to *NNF* formulas, and [LB08b] introduces a compact simultaneous representation of dependency sets for all universals in a formula.

Overall, we think that the amount of simultaneous and subsequent work suggests that variable dependencies are indeed a crucial topic in *QBF* reasoning that warrants further intensive research.

5.3. Universal Expansion Refinements

In the previous chapters, the expansion of a universal quantifier in a *QBF** formula has been defined as the process of duplicating the matrix of the formula and the dependent existentials:

$$\begin{aligned} Q\mathbf{v}\forall x\exists y_1 ... \exists y_m\ \phi(\mathbf{v}, x, y_1, ..., y_m, \mathbf{z}) \approx\ & Q\mathbf{v}\exists y_1^{(0)} ... \exists y_m^{(0)} \exists y_1^{(1)} ... \exists y_m^{(1)} \\ & \phi(\mathbf{v}, 0, y_1^{(0)}, ..., y_m^{(0)}, \mathbf{z}) \\ \wedge\ & \phi(\mathbf{v}, 1, y_1^{(1)}, ..., y_m^{(1)}, \mathbf{z}) \end{aligned}$$

We are now going to develop some refinements to this procedure to make it more powerful and more efficient in practice. While the results in this section (plus a further refinement in Section 5.6) are crucial to our preprocessing approach, they can also be applied independently in other contexts. For example, as mentioned in the previous section, [LB08a] applies some of our ideas to *NNF* formulas.

5.3.1. Selective Expansion

In existing applications of universal quantifier expansion for *QBF** formulas ([AB02, Bie05] and Chapter 3), universals have only been eliminated according to their nesting order, that is, the innermost universal quantifiers had to be expanded first. This is not so much of a problem when all universals are expanded anyway, be it step-wise and interleaved with other techniques as in [AB02, Bie05] or at once as in Chapter 3. However, in our preprocessing scenario in which we only want to eliminate selected individual universal quantifiers, it can have significant advantages to expand universals that are not in the innermost universal quantifier block:

- universals from other quantifier blocks may be cheaper to expand. Consider the example $\Phi = \forall x_1 \exists y_1 \forall x_2 \forall x_3 \exists y_2 \exists y_3 \forall x_4 \forall x_5 \exists y_4 \exists y_5\ (\phi \wedge \psi)$. If Φ can be rewritten as $\Phi' = \forall x_1 \exists y_1\ (\forall x_2 \exists y_2 \forall x_4 \exists y_4\ \phi) \wedge (\forall x_3 \exists y_3 \forall x_5 \exists y_5\ \psi)$, it is clear that we only have to duplicate ϕ when expanding x_2 and x_4, and analogously for ψ and x_3 and x_5. That means if $|\phi| < |\psi|$, it should indeed be cheaper to expand x_4 and x_2 before x_5 and x_3. Of course, that requires detecting the locality of the universals to avoid duplicating the whole formula matrix. This is discussed in more detail in the next section.

- some universal block other than the innermost might consist of only a few quantifiers. By expanding these, the number of quantifier blocks can be reduced and the formula may become much easier to solve. For example, if $\Psi = \exists y_1 \exists y_2 \forall x_1 \exists y_3 \forall x_2 \forall x_3 \forall x_4 \exists y_4 \exists y_5 \ \psi$ then we can either expand all innermost universals x_2, x_3 and x_4 to obtain a formula with only three quantifier alternations, or we can achieve the same effect by expanding only x_1.
- some universals from inner blocks might occur in short clauses, so that their expansion quickly leads to valuable unit literals or produces clauses that subsume longer clauses. Such strategic considerations will be covered in Section 5.5.

Extending universal expansion to arbitrary quantifier blocks is quite straightforward: all quantifier blocks that are further right in the prefix must be duplicated if they are existential and remain unchanged if they are universal.

Theorem 5.3.1. *(Selective Universal Quantifier Expansion)*
Let $\Phi(\mathbf{z}) = \forall X_1 \exists Y_1 ... \forall X_r \exists Y_r \ \phi(X_1, ..., X_r, Y_1, ..., Y_r, \mathbf{z})$ *with universal quantifier blocks* $X_k = (x_{k,1}, ..., x_{k,n_k})$ *and existential blocks* $Y_k = (y_{k,1}, ..., y_{k,m_k})$ *be a QBF* formula, and let the expansion of a universal quantifier* $\forall x_{i,j}$ *in* Φ *be the formula*

$$\begin{aligned}\Phi'(\mathbf{z}) \ = \ & Q' \ \phi(x_{1,1}, ..., x_{i,j-1}, 0, x_{i,j+1}, ..., x_{r,n_r}, y_{1,1}, ..., y_{i-1,m_{i-1}}, y^{(0)}_{i,1}, ..., y^{(0)}_{r,m_r}, \mathbf{z}) \\ & \wedge \ \phi(x_{1,1}, ..., x_{i,j-1}, 1, x_{i,j+1}, ..., x_{r,n_r}, y_{1,1}, ..., y_{i-1,m_{i-1}}, y^{(1)}_{i,1}, ..., y^{(1)}_{r,m_r}, \mathbf{z})\end{aligned}$$

with the new prefix Q' *that we obtain from the original prefix* Q *by dropping* $\forall x_{i,j}$ *and replacing* $\exists Y_k$ *with duplicates* $\exists Y^{(0)}_k, Y^{(1)}_k = \exists y^{(0)}_{k,1} ... \exists y^{(0)}_{k,m_k} \exists y^{(1)}_{k,1} ... \exists y^{(1)}_{k,m_k}$ *for all* $k = i, ..., r$. *Then* $\Phi(\mathbf{z}) \approx \Phi'(\mathbf{z})$.

Proof:
Let $Q_{\to x_{i,j}} := \forall X_1 \exists Y_1 ... \forall X_{i-1} \exists Y_{i-1} \forall x_{i,1} ... \forall x_{i,j-1}$ denote the beginning of the prefix Q up to, but excluding, $x_{i,j}$. Similarly, let $Q_{x_{i,j}\to}$ be the remainder of Q after $x_{i,j}$. Moreover, we abbreviate $\mathbf{x}|_{x_i=0} := x_{1,1}, ..., x_{i,j-1}, 0, x_{i,j+1}, ..., x_{r,n_r}$ and $\mathbf{x}|_{x_i=1} := x_{1,1}, ..., x_{i,j-1}, 1, x_{i,j+1}, ..., x_{r,n_r}$.
Applying $\forall x \ \Psi(x) \approx \Psi(0) \wedge \Psi(1)$ on x_i produces the following expansion:

$$\begin{aligned}\Phi(\mathbf{z}) \approx Q_{\to x_{i,j}} \ & (Q_{x_{i,j}\to} \ \phi(\mathbf{x}|_{x_i=0}, y_{1,1}, ..., y_{r,m_r}, \mathbf{z})) \\ & \wedge \ (Q_{x_{i,j}\to} \ \phi(\mathbf{x}|_{x_i=1}, y_{1,1}, ..., y_{r,m_r}, \mathbf{z}))\end{aligned}$$

We must restore the prenex form on the right hand side. This requires considering the sequence of quantifier blocks in $Q_{x_{i,j}\to}$. If $Q_{x_{i,j}\to} = \forall X_i' \exists Y_i ... \forall X_r \exists Y_r$ with a leading universal block $X_i' = (x_{i,j+1} ... x_{i,n_i})$, we can use the equivalence $(\forall x\, \Psi(x)) \wedge (\forall x\, \Gamma(x)) \approx \forall x\ (\Psi(x) \wedge \Gamma(x))$ on all universals in X_i' to move them to the front:

$$\begin{aligned}\Phi(\mathbf{z}) \approx Q_{\to x_{i,j}} \forall X_i'\ & (\exists Y_i ... \forall X_r \exists Y_r\ \phi(\mathbf{x}|_{x_i=0}, y_{1,1}, ..., y_{r,m_r}, \mathbf{z})) \\ \wedge\ & (\exists Y_i ... \forall X_r \exists Y_r\ \phi(\mathbf{x}|_{x_i=1}, y_{1,1}, ..., y_{r,m_r}, \mathbf{z}))\end{aligned}$$

In the other case $Q_{x_{i,j}\to} = \exists Y_i \forall X_{i+1} ... \exists Y_r$, we rename all the leading existentials $Y_i = (y_{i,1}, ..., y_{i,m_i})$ into $Y_i^{(0)} = (y_{i,1}^{(0)}, ..., y_{i,m_i}^{(0)})$ and $Y_i^{(1)} = (y_{i,1}^{(1)}, ..., y_{i,m_i}^{(1)})$:

$$\begin{aligned}\Phi(\mathbf{z}) \approx Q_{\to x_{i,j}}\ & (\exists Y_i^{(0)} \forall X_{i+1} ... \exists Y_r\ \phi(\mathbf{x}|_{x_i=0}, Y_1, ..., Y_{i-1}, Y_i^{(0)}, Y_{i+1}, ..., Y_r, \mathbf{z})) \\ \wedge\ & (\exists Y_i^{(1)} \forall X_{i+1} ... \exists Y_r\ \phi(\mathbf{x}|_{x_i=1}, Y_1, ..., Y_{i-1}, Y_i^{(1)}, Y_{i+1}, ..., Y_r, \mathbf{z}))\end{aligned}$$

Having distinct names allows us to move the existentials to the front:

$$\begin{aligned}\Phi(\mathbf{z}) \approx\ & Q_{\to x_{i,j}} \exists y_{i,1}^{(0)} ... \exists y_{i,m_i}^{(0)} \exists y_{i,1}^{(1)} ... \exists y_{i,m_i}^{(1)} \\ & (\forall X_{i+1} ... \exists Y_r\ \phi(\mathbf{x}|_{x_i=0}, Y_1, ..., Y_{i-1}, Y_i^{(0)}, Y_{i+1}, ..., Y_r, \mathbf{z})) \\ \wedge\ & (\forall X_{i+1} ... \exists Y_r\ \phi(\mathbf{x}|_{x_i=1}, Y_1, ..., Y_{i-1}, Y_i^{(1)}, Y_{i+1}, ..., Y_r, \mathbf{z}))\end{aligned}$$

The procedure continues inductively on $\exists Y_i ... \forall X_r \exists Y_r$ or $\forall X_{i+1} ... \exists Y_r$, respectively, until all quantifiers have been moved to the front. □

The proof shows that prenexing plays an important role in our generalized version of universal quantifier expansion. In the above formulation, our prenexing strategy is to keep both copies of an existential quantifier in the same block as the original quantifier. That means we merge the corresponding quantifier blocks from both subformulas, but we do not change the relative ordering of the quantifiers.

There are other prenexing approaches which are more aggressive in shifting quantifiers: [EST+04] provides a detailed comparison of different strategies for prenexing in general. However, in our unique scenario of universal expansion, our less intrusive prenexing that keeps the quantifier ordering intact has several advantages. On the one hand, it is very fast, because no further computations are necessary and because the duplication of dependent existentials can be performed in-place without moving around in the prefix. On the other hand, we

are not dealing with arbitrary non-prenex formulas, but with two copies of a formula that has already been in prenex form. Assuming that this prenexing has been carried out in an optimal way by the creator of the original formula, the expansion of a single universal will not provide much potential for further improvement. Finally, while the comparison in [EST+04] demonstrates that the choice of prenexing strategy can have a significant performance impact, there does not seem to exist a "silver bullet" strategy that performs best for the majority of solvers, and we do not want to specially tune-up our preprocessor towards individual solvers.

5.3.2. Variable Connectivity and Dependencies

Usually, not all clauses of a *QCNF** formula are affected by the expansion of a universal. An implementation of the algorithm suggested by the previous theorem will only have to touch clauses which contain either the expanded universal x itself or one of the existentials that must be duplicated. This observation implies that the size of the expansion depends in particular on the number of existential variables which are dominated by x. That makes it important to trim the set of these dominated existentials as much as possible.

The example formula $\Phi = \forall x_1 \exists y_1 \forall x_2 \forall x_3 \exists y_2 \exists y_3 \forall x_4 \forall x_5 \exists y_4 \exists y_5 \ (\phi \wedge \psi)$ from the preceding section that we have assumed to be the linearization of a non-prenex formula $\Phi' = \forall x_1 \exists y_1 \ (\forall x_2 \exists y_2 \forall x_4 \exists y_4 \ \phi) \wedge (\forall x_3 \exists y_3 \forall x_5 \exists y_5 \ \psi)$ clearly demonstrates that the ordering of the prefix alone provides only an upper bound for the set of dependent existentials: from the prefix in Φ, it appears that x_2 dominates all subsequent existentials $y_2, ..., y_5$, but by examining the matrix of the formula, we can discover that x_2 does not occur in common clauses with y_3 and y_5 due to the original non-prenex structure Φ'. The locality of x_2 therefore implies that y_3 and y_5 do not depend on x_2.

The idea of reducing variable dependencies by examining how variables are actually connected in common clauses has been introduced in [Bie05], but only for expansion from the innermost universal quantifier block. The paper informally defines a local connectivity relation between the expanded universal x and an existential y from the innermost existential quantifier block that holds if and only if both occur in the same clause. The set of dependent existentials is then given by the transitive closure of local connectivity to x, taking into account only x itself and the existentials from the innermost block.

For the expansion of universals from arbitrary quantifier blocks, we need to generalize the above definition of connectivity. In our formulation, connectivity is propagated by the expanded universal x and existentials from all quantifier blocks that occur further right in the prefix, but it is not propagated by other universal quantifiers.

Definition 5.3.2. *(Variable Connectivity and Dependent Existentials)*
*We denote a variable v **locally connected** to another variable w in a QCNF* formula if and only if both occur in a common clause. We then write $v \sim w$, and we let this relation $\sim$ be reflexive.*

Given a universal variable $x_{i,j}$ from the i-th universal quantifier block in a prefix of the form $Q = \forall X_1 \exists Y_1 ... \forall X_r \exists Y_r$, we define

$$\begin{aligned} D^{(0)}_{x_{i,j}} &:= \{y \in Y_i \cup ... \cup Y_r \mid y \sim x_{i,j}\} \\ D^{(k+1)}_{x_{i,j}} &:= \{y \in Y_i \cup ... \cup Y_r \mid y \sim y' \text{ for some } y' \in D^{(k)}_{x_{i,j}}\},\ k \geq 0 \\ D_{x_{i,j}} &:= \bigcup_k D^{(k)}_{x_{i,j}} \end{aligned}$$

*and call the set $D_{x_{i,j}}$ the **dependent existentials** of $x_{i,j}$.*

Consider the following example:

$$\begin{aligned} \Psi(\mathbf{z}) = \; & \forall x_{1,1} \forall x_{1,2} \exists y_{1,1} \exists y_{1,2} \forall x_{2,1} \exists y_{2,1} \exists y_{2,2} \\ & (x_{1,1} \vee x_{1,2} \vee \neg y_{1,1}) \wedge (x_{1,2} \vee y_{2,2}) \wedge \\ & (\neg x_{1,1} \vee y_{1,2}) \wedge (y_{1,1} \vee x_{2,1} \vee y_{2,1}) \wedge (y_{1,1} \vee \neg y_{2,1}) \end{aligned}$$

Then $D_{x_{1,1}} = \{y_{1,1}, y_{1,2}, y_{2,1}\}$, $D_{x_{1,2}} = \{y_{1,1}, y_{2,1}, y_{2,2}\}$ and $D_{x_{2,1}} = \{y_{2,1}\}$.

A simple algorithm for computing dependencies iteratively as in the above definition is provided in Listing 5.1. The algorithm first visits all clauses C that contain positive or negative literals over the given universal x and collects all existentials y with $x < y$ that occur in such a clause C. In the definition, the set of these existentials is called $D^{(0)}_x$. Then the process is repeated for all $y \in D^{(0)}_x$ by searching in unvisited clauses C with $y \in C$ or $\neg y \in C$ for more existentials in the scope of x, and so on. For a single universal, this algorithm needs time $O(|\Phi|)$ on a formula of length $|\Phi|$ that is given in a suitable *CNF* data structure [KBL99].

Listing 5.1: Computation of dependent existentials

```
Input Φ ∈ QCNF*, universal variable x in Φ;

D_x = ∅;
pending = {x};
while (pending ≠ ∅) {
  v = pending.removeElement();
  for each {C ∈ Φ | v ∈ C or ¬v ∈ C}
    if (C not marked as visited) {
      mark C as visited;
      for each {y ∈ y | (y ∈ C or ¬y ∈ C) and x < y}
        if (y not marked as seen) {
          mark y as seen;
          D_x = D_x ∪ {y};
          pending = pending ∪ {y};
        }
    }
}

Output Dependent existentials D_x as in Definition 5.3.2.
```

Rather than performing this computation repeatedly for each universal in the input formula, a very recent paper by Biere and Lonsing [LB08b] suggests an extension of these ideas that can reuse dependencies of one universal when computing dependencies of another universal, which leads to a simultaneous representation of dependencies for all universals in the formula.

Theorem 5.3.3. *Let $\Phi(\mathbf{z}) = Q\ \phi(\mathbf{z})$ be a formula in QCNF* that has the prefix $Q = \forall X_1 \exists Y_1 ... \forall X_r \exists Y_r$, and let $\forall x_{i,j}$ be a quantifier from the i-th universal block. Then Φ has an equivalence model $M = (f_{y_{1,1}}, ..., f_{y_{r,m_r}})$ such that $f_{y_{k,l}}$ does not depend on $x_{i,j}$ if $y_{k,l} \notin D_{x_{i,j}}$.*

Proof:
We only have to show that $f_{y_{k,l}}$ can be chosen independently of $x_{i,j}$ for every $y_{k,l} \in Y_i \cup ... \cup Y_r \setminus D_{x_{i,j}}$.
Let ϕ^D denote the conjunction of all clauses in ϕ which contain at least one variable in $D_{x_{i,j}}$. Without loss of generality, we assume that Φ is forall-reduced, so that all clauses with $x_{i,j}$ or $\neg x_{i,j}$ are also in ϕ^D. Let ϕ^I be the conjunction of the other clauses. Using the notation from the proof of Theorem 5.3.1 in the

previous section, we write $Q_{\to x_{i,j}} := \forall X_1 \exists Y_1 ... \forall X_{i-1} \exists Y_{i-1} \forall x_{i,1} ... \forall x_{i,j-1}$ for the beginning of the prefix Q up to, but excluding, $x_{i,j}$ and $Q_{x_{i,j}\to}$ for the remainder of Q after $x_{i,j}$. Then Φ has the form $\Phi = Q_{\to x_{i,j}} \forall x_{i,j} Q_{x_{i,j}\to}\ \phi^D \wedge \phi^I$.
We have $Q_{x_{i,j}\to} = Q' \exists Y_r$ with the innermost block being existential. Definition 5.3.2 implies that all existentials in ϕ^I are either bound in $Q_{\to x_{i,j}}$, or they only occur within clauses in ϕ^I. That means each existential quantifier $\exists y_{r,s}$ in $\exists Y_r$ belongs to one of two classes: either $y_{r,s} \in D_{x_{i,j}}$ and $y_{r,s}$ occurs only in ϕ^D, or $y_{r,s} \notin D_{x_{i,j}}$ and $y_{r,s}$ occurs only in ϕ^I. Accordingly, we can split $\exists Y_r$ into two distinct quantifier blocks $\exists Y_r^D$ and $\exists Y_r^I$ which we can distribute onto both subformulas ϕ^D and ϕ^I: $\Phi \approx Q_{\to x_{i,j}} \forall x_{i,j} Q'\ (\exists Y_r^D\ \phi^D) \wedge (\exists Y_r^I\ \phi^I)$.
Now $Q' = Q'' \forall X_r$ with innermost universal block X_r (without loss of generality, let $i < r$), and miniscoping $\forall x\ (\Psi(x) \wedge \Gamma(x)) \approx (\forall x\ \Psi(x)) \wedge (\forall x\ \Gamma(x))$ allows us to apply $\forall X_r$ to both subformulas:

$$\Phi \approx Q_{\to x_{i,j}} \forall x_{i,j} Q''\ (\forall X_r \exists Y_r^D\ \phi^D) \wedge (\forall X_r \exists Y_r^I\ \phi^I)$$

Continuing the above process of distributing quantifiers, we ultimately obtain a non-prenex formula $\Phi \approx Q_{\to x_{i,j}} \forall x_{i,j}\ (Q^D\ \phi^D) \wedge (Q^I\ \phi^I)$, and with $x_{i,j} \notin vars(\phi^I)$, it follows that $\Phi \approx Q_{\to x_{i,j}}\ (\forall x_{i,j} Q^D\ \phi^D) \wedge (Q^I\ \phi^I)$. Considering that existentials $y_{k,l} \in Y_i \cup ... \cup Y_r \setminus D_{x_{i,j}}$ occur only within ϕ^I, it is clear that such $y_{k,l}$ have equivalence model functions that do not depend on $x_{i,j}$. □

The proof illustrates that Definition 5.3.2 provides a way to recover non-prenex structure that is hidden in a prenex formula. We immediately have the following corollary:

Corollary 5.3.4. *When expanding a universal quantifier $\forall x_{i,j}$ from the i-th universal quantifier block ($1 \leq i \leq r$) of a forall-reduced QCNF* formula, we only need to duplicate (for $x_{i,j} = 0$ and $x_{i,j} = 1$) the dependent existential variables in $D_{x_{i,j}}$ and the clauses in which these existentials occur.*

This result has an interesting consequence for modeling problems in *QCNF**: universal expansion makes it relatively cheap to introduce locally used universal quantifiers. This allows natural encodings for various problems, in particular for those with an inherent non-prenex structure, while at the same time, our preprocessing can mitigate for the subsequent solver the overhead of having many universal variables. The QBFLIB formula collection [GNT01] includes several well-known families of encodings with large numbers of local universal quantifiers, e.g. the *Adder* suite [AB02] or the *Mneimneh-Sakallah* encodings [MS04].

5.3.3. Splitting Universal Scopes

In the preceding Definition 5.3.2, we obtain for a given universal $x_{i,j}$ the set of dependent existentials $D_{x_{i,j}}$ by a stepwise propagation of dependency. Starting with clauses that contain positive or negative $x_{i,j}$, dependency is extended to all variables in $Y_i \cup ... \cup Y_r$ that occur in the same clause, and then from these existentials to other existentials in $Y_i \cup ... \cup Y_r$ that they share a clause with, and so on. If we consider all quantified variables as vertices in a graph and let the local connectivity relation $\sim$ determine the graph's edges, every existential $u \in D_{x_{i,j}}$ is connected to $x_{i,j}$ by an undirected path $u \sim v_1$, $v_1 \sim v_2$, ..., $v_{k-1} \sim v_k$, $v_k \sim x_{i,j}$ with $v_1, ..., v_k \in D_{x_{i,j}}$.

Now if we consider a pair of two existentials $u, w \in D_{x_{i,j}}$, it is clear that they are connected to each other by a path from u to $x_{i,j}$ and then from $x_{i,j}$ to w. An interesting question is whether there also exists a path between u and w without passing through $x_{i,j}$. In general, this is not the case, as the simple example $\Phi = \forall x \exists y_1 \exists y_2 \exists y_3 \, (x \vee y_1) \wedge (\neg y_1 \vee y_2) \wedge (\neg x \vee y_3)$ shows. Here, $D_x = \{y_1, y_2, y_3\}$ and $y_2 \sim y_1$, $y_1 \sim x$ and $x \sim y_3$, such that y_2 and y_3 are connected by a path through x. But neither $y_1 \sim y_3$ nor $y_2 \sim y_3$, so there is no path between y_2 and y_3 without passing through x. We can, however, rewrite this example with the familiar miniscoping equivalence $\forall x \; (\Psi(x) \wedge \Gamma(x)) \approx (\forall x \, \Psi(x)) \wedge (\forall x \, \Gamma(x))$ that we have already used extensively in the previous sections. Then we have $\Phi \approx (\forall x \exists y_1 \exists y_2 \; (x \vee y_1) \wedge (\neg y_1 \vee y_2)) \wedge (\forall x \exists y_3 \; (\neg x \vee y_3))$, which we can rewrite as $\Phi \approx \forall x^{(1)} \exists y_1 \exists y_2 \forall x^{(2)} \exists y_3 \; (x^{(1)} \vee y_1) \wedge (\neg y_1 \vee y_2) \wedge (\neg x^{(2)} \vee y_3)$. Obviously, $D_{x^{(1)}} = \{y_1, y_2\}$, $D_{x^{(2)}} = \{y_3\}$ and $y_1 \sim y_2$, so all existentials in the same dependency set are connected without passing through the corresponding universal.

We now show that we can always achieve this property by splitting universal scopes with the above miniscoping $\forall x \; (\Psi(x) \wedge \Gamma(x)) \approx (\forall x \, \Psi(x)) \wedge (\forall x \, \Gamma(x))$. We start with a formal definition and some basic results.

Definition 5.3.5. *(Existentially Connected Dependencies)*
For a universal quantifier $\forall x$ in a QCNF* formula Φ, let D_x be the set of dependent existentials as in Definition 5.3.2. Then we call two variables $u, w \in D_x$ ***existentially connected with respect to*** *D_x if and only if $u \sim w$ or there exist $y_{i_1}, ..., y_{i_k} \in D_x$ with $u \sim y_{i_1}$, $y_{i_1} \sim y_{i_2}$, ..., $y_{i_{k-1}} \sim y_{i_k}$ and $y_{i_k} \sim w$ in Φ. We write $u \sim^*_{D_x} w$ for such variables u and w that are connected within D_x.*
We say that D_x itself is ***existentially connected*** *if and only if every pair $u, w \in D_x$ is existentially connected in D_x.*

If $u \sim^*_{D_x} v$ and $v \sim^*_{D_x} w$ within the same dependency set D_x, it clearly follows that $u \sim^*_{D_x} w$. With the reflexivity and symmetry of the underlying connectivity relation $\sim$, we immediately have the following proposition.

Proposition 5.3.6. *For all dependency sets D_x, existential connectivity $\sim^*_{D_x}$ is an equivalence relation.*

A useful property of existentially connected dependency sets is that two such sets in a *QCNF** formula are either disjoint, or one is entirely contained within the other.

Lemma 5.3.7. *Let $\Phi(\mathbf{z}) = \forall X_1 \exists Y_1 ... \forall X_r \exists Y_r \, \phi(X_1, ..., X_r, Y_1, ..., Y_r, \mathbf{z})$ be a QCNF* formula, and let $\forall x_{i,j}$ and $\forall x_{k,l}$ be a pair of universal quantifiers from quantifier blocks X_i and X_k with $i \leq k$ and dependent existentials $D_1 := D_{x_{i,j}}$ and $D_2 := D_{x_{k,l}}$.*
If D_1 is partitioned by $\sim^*_{D_1}$ into equivalence classes $D_1^{(1)}, ..., D_1^{(p)}$ ($p = 1$ if D_1 is fully existentially connected) and D_2 is existentially connected, we have

$$D_1 \cap D_2 = \emptyset \text{ or } D_2 \subseteq D_1^{(s)} \text{ for some } 1 \leq s \leq p \, .$$

Proof:
For $|D_2| \leq 1$, the result is trivially true. Otherwise, let $u, w \in D_2$, $u \neq w$, be a pair of distinct existentials in D_2. If $u, w \notin D_1$ for every such pair, we have $D_1 \cap D_2 = \emptyset$. Else let $u \in D_1^{(s)}$ without loss of generality, and we must show that $w \in D_1^{(s)}$ as well. Since D_2 is existentially connected, there exist $v_1, ..., v_e \in D_2$ with $u \sim v_1$, $v_1 \sim v_2$, ..., $v_{e-1} \sim v_e$ and $v_e \sim w$. From $i \leq k$, it follows that these existentials $v_1, ..., v_e$ are also in the scope of $\forall x_{i,j}$. With $u \in D_1$ and the above chain of connectivity, if follows from Definition 5.3.2 that $\{w, v_1, ..., v_e\} \subseteq D_1$. Then Definition 5.3.5 implies that u and w are existentially connected $u \sim^*_{D_1} w$ within D_1, and thus $u, w \in D_1^{(s)}$, because $D_1^{(s)}$ is an equivalence class of D_1 with respect to $\sim^*_{D_1}$. □

Theorem 5.3.8. *For every QCNF* formula $\Phi(\mathbf{z})$, there exists an equivalent formula $\Phi'(\mathbf{z}) \in$ QCNF* in which the dependency sets $D_{x_{i,j}}$ are existentially connected for every universal quantifier $\forall x_{i,j}$ in Φ', and Φ' can be computed deterministically in polynomial time.*

Notice that without the requirement that Φ' can be computed in deterministic polynomial time and therefore has polynomial length, the theorem could be fulfilled in a trivial way by simply expanding all universal quantifiers in Φ.

Proof:
The proof is by induction on the number of universal quantifiers in the original formula. It is based on the idea of splitting universal scopes that are not existentially connected by duplicating the corresponding quantifiers as in the example at the beginning of this section.

For the induction base, let $\Phi(\mathbf{z}) = \forall x \exists y_1 ... \exists y_m \ \phi(x, y_1, ..., y_m, \mathbf{z})$ be a formula with a single universal. Without loss of generality, we assume that $\forall x$ is the outermost quantifier, because leading existentials can simply be considered as free variables. If the dependency set D_x is not existentially connected, $\sim^*_{D_x}$ partitions D_x into existentially connected equivalence classes $D_x^{(1)}, ..., D_x^{(k)}$ with $k \geq 2$. Then every clause in ϕ contains existentials from at most one of these partitions, and we can also partition ϕ into subformulas $\phi^{(0)}, ..., \phi^{(k)}$: for $1 \leq i \leq k$, $\phi^{(i)}$ is the conjunction of all clauses that have at least one literal over existentials in $D_x^{(i)}$, and $\phi^{(0)}$ collects all clauses without any literals over existentials in D_x. Let $\{y_1^{(i)}, ..., y_{m_i}^{(i)}\} := D_x^{(i)}$ and $\{y_1^{(0)}, ..., y_{m_0}^{(0)}\} := \{y_1, ..., y_m\} \setminus D_x$, then we have

$$\Phi(\mathbf{z}) \approx \forall x \bigwedge_{i=0...k} (\exists y_1^{(i)} ... \exists y_{m_i}^{(i)} \ \phi^{(i)}(x, y_1^{(i)}, ..., y_{m_i}^{(i)}, \mathbf{z})) \ .$$

With $\forall x \ (\Psi(x) \wedge \Gamma(x)) \approx (\forall x \, \Psi(x)) \wedge (\forall x \, \Gamma(x))$, we finally obtain:

$$\begin{aligned} \Phi(\mathbf{z}) \ &\approx \ \textstyle\bigwedge_{i=0...k} (\forall x \exists y_1^{(i)} ... \exists y_{m_i}^{(i)} \ \phi^{(i)}(x, y_1^{(i)}, ..., y_{m_i}^{(i)}, \mathbf{z})) \\ &\approx \ \forall x^{(0)} ... \forall x^{(k)} \exists y_1^{(0)} ... \exists y_{m_k}^{(k)} \ \textstyle\bigwedge_{i=0...k} \phi^{(i)}(x^{(i)}, y_1^{(i)}, ..., y_{m_i}^{(i)}, \mathbf{z}) \end{aligned}$$

Clearly, $D_{x^{(i)}} = D_x^{(i)}$ for $1 \leq i \leq k$ by construction. In addition, $D_{x^{(0)}} = \emptyset$, so we might discard $x^{(0)}$ by universal reduction. The formula above can easily be computed in polynomial time: for every clause over positive or negative x, we compute a dependency set as in Definition 5.3.2, but without considering any other clauses over x. Let l be the number of clauses over x, then we obtain dependency sets $\widetilde{D}_1, ..., \widetilde{D}_l$, and for all $1 \leq j \leq l$, we have $\widetilde{D}_j = D_x^{(i)}$ for some $1 \leq i \leq k$, which means we find (possibly multiple copies of) the existentially connected partitions $D_x^{(i)}$ of D_x.

For the induction step, let $\Phi(\mathbf{z}) = \forall x Q\ \phi(x,\mathbf{z})$ be a formula where $\forall x$ is the outermost universal quantifier. By the induction hypothesis, $Q\ \phi(x,\mathbf{z})$ is equivalent to some $Q'\ \phi'(x,\mathbf{z}) \in QCNF^*$ with existentially connected dependencies. Then $\Phi(\mathbf{z}) \approx \forall x Q'\ \phi'(x,\mathbf{z})$, and if D_x is not existentially connected in $Q'\ \phi'$, we partition D_x as above into equivalence classes $D_x^{(1)},...,D_x^{(k)}$ and let $D_x^{(0)}$ contain all other existentials in Q' that are not in D_x. From Lemma 5.3.7, it now follows that for every universal x' in Q', we either have $D_{x'} \cap D_x = \emptyset$ or $D_{x'} \subseteq D_x^{(i)}$ for some i with $1 \leq i \leq k$. This allows us to partition $Q'\ \phi'$ as in the induction base. For $1 \leq i \leq k$, we let $Q'^{(i)}$ be the subsequence of Q' that contains only those existential quantifiers that are referenced in $D_x^{(i)}$ and those universal quantifiers $\forall x'$ with $D_{x'} \subseteq D_x^{(i)}$. Similarly, $Q'^{(0)}$ contains those existentials that belong to $D_x^{(0)}$ (i.e. those that are not in D_x) and those universals $\forall x'$ with $D_{x'} \cap D_x = \emptyset$. Due to the aforementioned Lemma 5.3.7, every quantifier in Q' belongs to exactly one subsequence $Q'^{(i)}$, and for every universal quantifier $\forall x'$ in a subsequence $Q'^{(i)}$, all existentials $D_{x'}$ that it dominates are also in the same subsequence $Q'^{(i)}$. Thus

$$\Phi(\mathbf{z}) \approx \forall x \bigwedge_{i=0...k} \left(Q'^{(i)}\ \phi'^{(i)}(x,\mathbf{z}) \right)$$

where we denote by $\phi'^{(i)}$ the conjunction of all clauses in ϕ' that contain at least one literal over existentials in $D_x^{(i)}$. The remainder of the proof is now the same as in the induction base:

$$\Phi(\mathbf{z}) \approx \forall x^{(0)}...\forall x^{(k)} Q' \bigwedge_{i=0...k} \phi'^{(i)}(x^{(i)},\mathbf{z})$$

by distributing $\forall x$ over the conjunction, so that it is split up into k distinct universal quantifiers that have existentially connected dependencies each. □

The algorithm outlined in the proof leaves the duplicated universal quantifiers at the same positions as the original quantifiers, and the existential blocks are not modified at all. With Lemma 5.3.7, we obtain the following corollary.

Corollary 5.3.9. *For every $\Phi(\mathbf{z}) = \forall X_1 \exists Y_1 ... \forall X_r \exists Y_r\ \phi(\mathbf{x},\mathbf{y},\mathbf{z}) \in QCNF^*$, it is possible to compute in deterministic polynomial time an equivalent $QCNF^*$ formula $\Phi'(\mathbf{z}) = \forall X'_1 \exists Y_1 ... \forall X'_r \exists Y_r\ \phi'(\mathbf{x}',\mathbf{y},\mathbf{z})$, such that*

$$D_{x'_{i,j}} \cap D_{x'_{k,l}} = \emptyset \text{ or } D_{x'_{k,l}} \subseteq D_{x'_{i,j}}$$

for every pair of universal quantifiers $\forall x'_{i,j}$ and $\forall x'_{k,l}$ from quantifier blocks X'_i and X'_k with $i \leq k$.

Before we present an example and discuss the meaning of this result, we would like to have a look at a possible implementation of such a transformation from $\Phi \in QCNF^*$ to $\Phi' \in QCNF^*$ with existentially connected dependencies. The actual task is to find the existentially connected partitions of the dependent existentials for each universal variable x. Listing 5.2 shows how the original dependency computation from Listing 5.1 that was presented in Section 5.3.2 can be slightly modified, so that the reachable existentials are computed separately for each single clause over x or $\neg x$. The trick is to check whether such a clause has already been reached from another clause over x or $\neg x$ earlier in the process. If this is the case, both clauses must belong to the same existentially connected partition of D_x, and a repeated search for reachable existentials is unnecessary. That way, it is still sufficient to inspect every clause in the input formula Φ at most once for each universal x in Φ. In total, $O(n \cdot |\Phi|)$ clauses are visited to perform this transformation on a formula of length $|\Phi|$ with n universal quantifiers.

Consider the following example:

$$\begin{aligned}\Psi(\mathbf{z}) \;=\; & \forall x_{1,1} \forall x_{1,2} \exists y_{1,1} \forall x_{2,1} \exists y_{2,1} \exists y_{2,2} \exists y_{2,3} \\ & (x_{1,1} \vee x_{1,2} \vee \neg y_{1,1}) \wedge (x_{1,2} \vee y_{2,3}) \wedge (y_{1,1} \vee \neg x_{2,1} \vee y_{2,1}) \\ & (\neg x_{1,1} \vee y_{2,2}) \wedge (y_{1,1} \vee x_{2,1} \vee \neg y_{2,2})\end{aligned}$$

Here, $x_{1,1}$ occurs within two clauses. The algorithm starts by considering only the first one and includes $y_{1,1}$ in the set of dependent variables. The occurrences of $y_{1,1}$ lead to the third and fifth clause, which adds $y_{2,1}$ and $y_{2,2}$. Following these variables does not reveal any additional dependent existentials, so we obtain:

$$D^{(1)}_{x_{1,1}} = \{y_{1,1}, y_{2,1}, y_{2,2}\}$$

Now, we consider the second occurrence of $x_{1,1}$, which is in the fourth clause $(\neg x_{1,1} \vee y_{2,2})$. But $y_{2,2}$ has already been found in the previous step, so there is no second partition of $x_{1,1}$. For $x_{1,2}$, starting with the first clause also yields:

$$D^{(1)}_{x_{1,2}} = \{y_{1,1}, y_{2,1}, y_{2,2}\}$$

$x_{1,2}$ also occurs in the second clause, which is not reachable from the first clause:

$$D^{(2)}_{x_{1,2}} = \{y_{2,3}\}$$

In the last step, the occurrence of $x_{2,1}$ in the third clause leads to

$$D^{(1)}_{x_{2,1}} = \{y_{2,1}\} \,.$$

Listing 5.2: Computation of dependencies with existential connectivity

```
Input Forall-reduced Φ ∈ QCNF*, universal variable x in Φ;

k = 0;
for each {C ∈ Φ | x ∈ C or ¬x ∈ C}
  if (C not marked as visited) {
    mark C as visited;
    k++;
    D_x^(k) = ∅;
    pending = ∅;
    for each {y ∈ y | (y ∈ C or ¬y ∈ C) and x < y} {
      mark y as seen;
      pending = pending ∪ {y};
    }
    while (pending ≠ ∅) {
      v = pending.removeElement();
      D_x^(k) = D_x^(k) ∪ {v};
      for each {C ∈ Φ | v ∈ C or ¬v ∈ C}
        if (C not marked as visited) {
          mark C as visited;
          for each {y ∈ y | (y ∈ C or ¬y ∈ C) and x < y}
            if (y not marked as seen) {
              mark y as seen;
              pending = pending ∪ {y};
            }
        }
    }
  }

Output Dependent existentials in partitions D_x^(1),...,D_x^(k).
```

Notice that $y_{1,1}$ is not in the scope of $x_{2,1}$. From the fifth clause, we get:

$$D^{(2)}_{x_{2,1}} = \{y_{2,2}\}$$

Consequently, the original formula is equivalent to the following:

$$\begin{aligned}\Psi'(\mathbf{z}) \;=\; & \forall x^{(1)}_{1,1}\forall x^{(1)}_{1,2}\forall x^{(2)}_{1,2}\exists y_{1,1}\forall x^{(1)}_{2,1}\forall x^{(2)}_{2,1}\exists y_{2,1}\exists y_{2,2}\exists y_{2,3}\\ & (x^{(1)}_{1,1}\vee x^{(1)}_{1,2}\vee\neg y_{1,1})\wedge(x^{(2)}_{1,2}\vee y_{2,3})\wedge(y_{1,1}\vee\neg x^{(1)}_{2,1}\vee y_{2,1})\\ & (\neg x^{(1)}_{1,1}\vee y_{2,2})\wedge(y_{1,1}\vee x^{(2)}_{2,1}\vee\neg y_{2,2})\end{aligned}$$

In addition, it would be possible reorder the prefix of Ψ', e.g. by moving $\forall x^{(2)}_{1,2}$ further to the right, but we will later see that for our preprocessing purposes, it is usually best to keep the existing ordering.

Corollary 5.3.9 can be understood as follows: it implies that the dependencies in Φ' can be represented as a *dependency tree* in which $Y = Y_1 \cup ... \cup Y_r$ is the root and the other nodes are the dependency sets for the universal variables in Φ'. Let the first index of a universal indicate the number of the quantifier block. Then the tree can be constructed in such a way that a node $u = D_{x'_{k,l}}$ is a child of another node $v = D_{x'_{i,j}}$ if and only if (i,j) is the largest index[1] with $(i,j) < (k,l)$ and $D_{x'_{k,l}} \subseteq D_{x'_{i,j}}$. All nodes without parent are connected to the root. For the example from the last paragraph, we obtain the dependency tree shown in Figure 5.1.

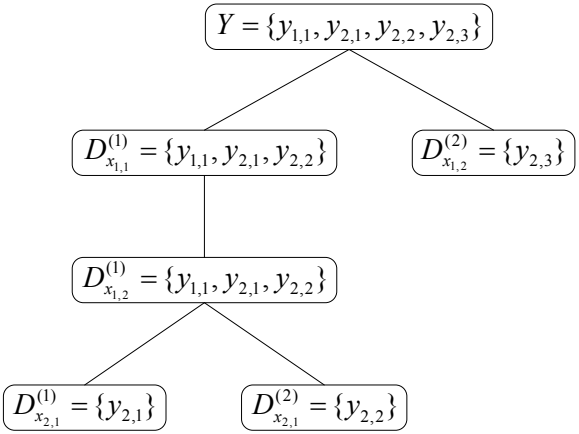

Figure 5.1.: Example of a dependency tree after splitting universal scopes

In [Ben05b], Benedetti has presented an algorithm to extract from a *QCNF* formula Φ a tree-shaped prefix, called a quantifier tree, to recover structural information that might have been lost during the prenexing of Φ. Interestingly, it is possible to extract from these quantifier trees the same dependency information as with our splitting of universal scopes. The reason is that both approaches are based on miniscoping $\forall x \ (\Psi(x) \wedge \Gamma(x)) \approx (\forall x \, \Psi(x)) \wedge (\forall x \, \Gamma(x))$, but this

[1] We define $(i,j) < (k,l)$ to hold if and only if either $i < k$ or $i = k, j < l$.

is applied in very different ways in the two algorithms, which makes the similarity difficult to see: Benedetti starts with individual existentially quantified variables and gradually merges their scopes, creating corresponding universal nodes as required. Our approach works the other way round by starting with occurrences of universal variables, from which it then propagates dependency to locally connected existentials. We claim that the latter direction from universals to existentials is more suitable for universal expansion. In particular, we do not have to build the whole quantifier tree to compute the dependencies for a single universal before expanding it.

Most importantly, we will present in Section 5.6 a powerful refinement of local connectivity that takes into account variable polarity and can thus reduce the cardinality of dependencies by an arbitrarily large factor in the best case. While this idea extends our approach in a very natural manner, we do not see an obvious way to combine it with Benedetti's quantifier trees. As an additional benefit, our work provides formal proofs, which are not given in [Ben05b].

We have already pointed out that Corollary 5.3.9 has the helpful property of leaving the structure of the prefix intact, so we can undo universal splits after the preprocessing is finished and merge the remaining partitions of universal quantifier scopes back into one common universal scope. This guarantees that the preprocessing will never increase the number of universal quantifiers in a formula. Splitting of universal scopes is particularly beneficial in cases where all remaining universals have very large dependencies and are therefore too expensive to expand as a whole. Then we can try to split universals into partitions that are small enough for expansion and/or contain lots of short clauses that provide valuable unit literals after an expansion. But the number of universals is only guaranteed to decrease when all partitions of a universal scope are expanded. That means we should first try to expand universal scopes as they are without splitting. This is no problem with our approach, since we can freely choose between the algorithm without existential connectivity from Listing 5.1 (Page 137) and the one from Listing 5.2 above that considers existential connectivity. In Benedetti's quantifier tree algorithm on the other hand, universals with multiple existentially connected partitions of dependent existentials are always duplicated by miniscoping.

5.4. Bounded Expansion for Preprocessing

Even when the locality of universals is observed and all dependencies are reduced accordingly, repeated application of universal expansion can easily lead to rapid formula growth. The idea of our preprocessing is to perform universal expansion as often as possible within a given upper bound for the growth of the formula. More precisely, for a constant bound $\beta_{exp} \geq 1$, we attempt to keep the length of the expanded formula below $\beta_{exp} \cdot |\Phi|$, where Φ is the original input formula (after some initial simplifications as described below). Many state-of-the-art *QBF** solvers appear to be very sensitive to the length of formulas, even in cases where longer formulas are structurally easier than shorter ones (see the experimental results in Section 5.7), so it is probably a good idea to allow the preprocessing output to grow only by a rather small factor. In our experiments, we have found values of $\beta_{exp} \approx 2$ to work well.

Another reason for choosing tight values of β_{exp} is that we want to apply universal expansion only as long as it appears worthwhile, that means as long as expanding a universal in the formula appears to be cheaper than the effort of handling that universal in the subsequently called solver. We attempt to achieve this by first expanding those universals which are estimated to be cheap to expand (see Section 5.5), and we terminate the preprocessing when there are no more universals left with estimated expansion costs low enough to stay below the threshold. If that threshold is low, universals that are expensive to expand will never be chosen, because the bound will already have been reached after expanding some of the cheaper universals. Listing 5.3 shows the basic structure of the preprocessor's main loop and illustrates where the expansion bound is applied. More details on the variable selection will be given in the next section. For completeness, we have also included the reduction of dependencies by resolution, which will be discussed in Section 5.6.

In Section 2.4, we have described various techniques for simplifying quantified Boolean formulas. Expanding a universally quantified variable may open up possibilities for such simplifications. For example, expanding x in a clause $(x \vee y \vee \mathbf{w})$ containing an existential y and some other literals $\mathbf{w}$ can turn y into a pure literal if $x = 1$ and this clause is the only one in which y occurs positively. Or if $\mathbf{w} = \emptyset$, which means the clause is in fact binary, we obtain a new unit literal y in the case that $x = 0$. By performing these simplifications, we can reduce the actual costs of universal expansion. We have thus included the

Listing 5.3: Bounded expansion preprocessing: main loop

```
Input Φ ∈ QCNF* and expansion bound β_exp ≥ 1;

simplify Φ;
Φ_exp = Φ;
expansionPossible = true;
while (expansionPossible) {
  choose universal x with smallest predicted costs c_x;
  if ((x ≠ null) && (|Φ_exp| + c_x ≤ β_exp · |Φ|)) {
    reduce dependencies D_x by resolution;
    expand x in Φ_exp according to dependencies D_x;
    simplify Φ_exp;
  } else expansionPossible = false;
}

Output Preprocessed formula Φ_exp ≈ Φ;
```

standard simplifications from Section 2.4 that are typically performed by *QBF** solvers: unit propagation, pure literal elimination, universal reduction, detection of dual binary clauses and subsumption checking. In Section 5.6, we will add the elimination of existentially quantified variables by Q-resolution to this tool set.

We apply the simplifications in a circular fashion where one simplification rule may trigger the application of another rule, until we reach closure. Initially, we attempt to simplify the whole input formula. Later, we check for specific simplifications as necessary. For the initial simplification, universal reduction is probably the most important operation and allows us to assume for the remaining process that all clauses are cleansed from trailing universal variables which do not dominate any existentials in the same clause. Whenever we later modify clauses or add new ones, we will make sure they are forall-reduced as well.

5.5. Variable Selection

The results from Section 5.3 give us lots of possibilities as to what we can expand: universal quantifiers from the whole prefix can be chosen, and even existentially connected partitions of a single universal scope can be expanded in-

dividually. But at the same time, the bound from the previous section must be observed in order to prevent excessive formula growth, which makes it usually impossible to expand all universals. It is thus crucial to the success of the preprocessing to make good choices for the universals to be expanded.

5.5.1. Expansion Costs and Selection Strategy

An obvious measurement to determine which expansions to make is to consider the possible growth of the formula. Each expansion of a universal x_i produces expansion costs $c_{x_i} = |\Phi'| - |\Phi|$, where $|\Phi|$ is the size of the current formula before the expansion and $|\Phi'|$ the size of the resulting formula. When considering an expansion sequence of universals $x_{i_1}, ..., x_{i_k}$, we face the problem that the costs for the last step are defined with respect to the formula that results from expanding $x_{i_1}, ..., x_{i_{k-1}}$ first. This makes the expansion of x_{i_k} very hard to predict without actually performing the whole sequence of preceding steps. The reason is that multiple expansions are not independent of each other: expanding one universal x_i might require duplicating clauses that are also in the dependency scope of another universal x_j, so that the costs of expanding x_j increase due to the prior expansion of x_i. Consider, for example, a formula of the form

$$\Phi = Q \forall x_i \exists y_k \forall x_j \exists y_l \widetilde{Q}\ \varphi \wedge (x_i \vee y_k \vee w) \wedge (y_k \vee y_l) \wedge (x_j \vee y_l \vee \widetilde{w})$$

where Q and $\widetilde{Q}$ are arbitrary sequences of quantifiers, φ is some subformula, and w and $\widetilde{w}$ are disjunctions of some literals. Then $y_k, y_l \in D_{x_i}$, so the clause $(y_k \vee y_l)$ must be duplicated into $(y_k^{(0)} \vee y_l^{(0)})$ and $(y_k^{(1)} \vee y_l^{(1)})$ when x_i is expanded. But $y_l^{(0)}$ and $y_l^{(1)}$ are then in D_{x_j}, so when x_j is expanded, we end up with four copies of this clause: $(y_k^{(0)} \vee y_l^{(0,0)})$, $(y_k^{(0)} \vee y_l^{(0,1)})$, $(y_k^{(1)} \vee y_l^{(1,0)})$ and $(y_k^{(1)} \vee y_l^{(1,1)})$. Had we only expanded x_j, but not x_i, there would have been only two copies of the clause, which means the expansion costs $c_{x_j|x_i}$ of expanding x_j after having already expanded x_i are more expensive by at least four literals (two copies of the binary clause). In the example, the same is also true for expanding x_i after x_j. Instead of expanding both x_i and x_j, it might have been better to only expand the cheaper of the two, say x_i, but then a different universal x_q. That one could even be more expensive than x_j, that means $c_{x_j} < c_{x_q}$, but the combined costs might still be lower, so that $c_{x_q|x_i} < c_{x_j|x_i}$.

The problem of mutual dependencies between multiple expansion steps is aggravated by the influence of simplifications. We have seen in the last section

that universal expansion can produce unit and pure literals, allow subsumption of clauses, etc. Clearly, these simplifications can significantly affect subsequent expansions, but they are also very hard to predict in advance due to the nature of (quantified) Boolean satisfiability problems. This makes it very unlikely that there is an efficient algorithm for determining the actual costs of expanding a series of multiple quantifiers without really performing all those expansions and the corresponding simplifications. Consequently, we cannot expect to find an optimal expansion schedule. This is in particular unavoidable if we consider that our preprocessor should typically consume much less computing time than a subsequently invoked full-blown solver.

[Bie05] suggests to schedule expansions according to a greedy heuristic that picks the universal with the lowest one-step expansion costs. This universal is then expanded and the resulting simplifications are carried out, and then again, the universal with the lowest costs is determined, etc. By making choices in such a stepwise fashion, it is no longer necessary to consider combined expansion costs $c_{x_i|x_{j_1},\ldots,x_{j_k}}$. Intuitively, this appears to be a reasonable strategy, although no further justification is given in the paper. We now show that this greedy heuristic does indeed produce schedules that are optimal if we exclude simplifications and consider only formulas with existentially connected dependencies. The latter requirement can be satisfied in polynomial time for all *QCNF** formulas by virtue of Theorem 5.3.8.

In the following, we use some helpful notation to describe occurrences of variables and literals. For given $\Phi \in QCNF^*$, we let $C(l) := \{C \mid C \in \Phi \text{ and } l \in C\}$ denote the set of clauses that contain the literal l. Similarly, for a variable v, we let $C^{\pm}(v) := C(v) \cup C(\neg v)$ be the set of clauses that contain a positive or negative literal over v. Furthermore, we let $C(L) := \bigcup_{l \in L} C(l)$ for sets of literals and $C^{\pm}(V) := \bigcup_{v \in V} C^{\pm}(v)$ for sets of variables. Let $\mathscr{C}$ be a set of clauses, then $\Sigma\mathscr{C} := \Sigma_{\gamma \in \mathscr{C}}|\gamma|$ denotes the sum of the sizes of all clauses in $\mathscr{C}$.
Consider the example formula $\Phi = \forall x_1 \exists y_1 \exists y_2\ (x_1 \vee \neg y_2) \wedge (\neg y_2 \vee y_1) \wedge \neg y_1$. Then we have, e.g., $\Sigma C(y_1) = 2$ and $\Sigma C^{\pm}(y_1) = 3$. Also, $\Sigma C^{\pm}(\{y_1, y_2\}) = 5$, because $C^{\pm}(\{y_1, y_2\}) = \{(x_1 \vee \neg y_2), (\neg y_2 \vee y_1), (\neg y_1)\}$.
Finally, for a set U of universal variables, we let

$$dup_\Phi(U) := \sum_{C \in \Phi} |C| \cdot \left(2^{|\{u \mid u \in U, C \in C^{\pm}(D_u)\}|} - 1\right)$$

be the costs of duplicating in a *QCNF** formula Φ every clause $C \in \Phi$ as often as it is dominated by one of the universals in U (cf. Corollary 5.3.4).

Theorem 5.5.1. *A greedy algorithm can determine for every $\Phi \in QCNF^*$ with universal quantifiers $\forall x_1,...,\forall x_n$ and existentially connected dependencies a set $\{x_{i_1},...,x_{i_k}\}$ with maximum k such that*

$$dup_\Phi(\{x_{i_1},...,x_{i_k}\}) \leq \beta \cdot |\Phi|$$

for a given constant β. In the l-th step ($l \geq 1$), the universal $x_{i_l} \in \{x_1,...,x_n\} \setminus \{x_{i_1},...,x_{i_{l-1}}\}$ that has the lowest weight $w(x_{i_l}) := dup_\Phi(\{x_{i_l}\} \cup \{x_{i_1},...,x_{i_{l-1}}\})$ is chosen. If $w(x_{i_l})$ is smaller than the bound, x_{i_l} is added to the solution, and another step is performed. Otherwise, $\{x_{i_1},...,x_{i_{l-1}}\}$ is the final solution.

Proof:
Consider an optimal solution $\{x_{j_1},...,x_{j_k}\}$, and let $x_{i_1},...,x_{i_k}$ be the universals chosen in this order by the above greedy algorithm. We now show by induction that all the elements in the given solution can be replaced with the elements computed by the greedy algorithm, without losing optimality.
Suppose $x_{i_1} \notin \{x_{j_1},...,x_{j_k}\}$. According to Lemma 5.3.7, each pair of universal quantifiers $\forall u$ and $\forall v$ (with $\forall u$ preceding $\forall v$ in the prefix) in the given formula has either disjoint or nested dependencies, that is $D_u \cap D_v = \emptyset$ or $D_v \subseteq D_u$. That means the first case for our proof is $D_{x_{i_1}} \cap D_{x_{j_1}} = \emptyset, ..., D_{x_{i_1}} \cap D_{x_{j_k}} = \emptyset$. Then we can replace x_{j_1} with x_{i_1} and still have an optimal solution:

$$dup_\Phi(\{x_{i_1},x_{j_2},...,x_{j_k}\}) = dup_\Phi(\{x_{i_1}\}) + dup_\Phi(\{x_{j_2},...,x_{j_k}\}) \leq$$
$$dup_\Phi(\{x_{j_1}\}) + dup_\Phi(\{x_{j_2},...,x_{j_k}\}) \leq dup_\Phi(\{x_{j_1},...,x_{j_k}\})$$

If $D_{x_{i_1}} \subseteq D_{x_{j_l}}$ for some $l \in \{1,...,k\}$, it is easy to see that

$$dup_\Phi(\{x_{j_1},...,x_{j_k}\} \setminus \{x_{j_l}\} \cup \{x_{i_1}\}) \leq dup_\Phi(\{x_{j_1},...,x_{j_k}\}) \ .$$

Finally, the case $D_{x_{i_1}} \supset D_{x_{j_l}}$ can only occur if $C^{\pm}(D_{x_{i_1}}) = C^{\pm}(D_{x_{j_l}})$, because $w(x_{i_1}) = dup_\Phi(x_{i_1}) \leq w(x_{j_l}) = dup_\Phi(x_{j_l})$. Then it is also clear that x_{j_l} can be replaced with x_{i_1}.
Now assume we have an optimal solution $\{x_{i_1},...,x_{i_r},x_{j_{r+1}},...,x_{j_k}\}$ which does not contain $x_{i_{r+1}}$. If $D_{x_{i_{r+1}}} \subseteq D_{x_{j_l}}$ for some $l \in \{r+1,...,k\}$, we can clearly replace x_{j_l} with $x_{i_{r+1}}$ without losing optimality. And if $D_{x_{i_{r+1}}} \supset D_{x_{j_l}}$, we have $C^{\pm}(D_{x_{i_{r+1}}}) = C^{\pm}(D_{x_{j_l}})$ as above, so we can also safely make the substitution. Otherwise, $D_{x_{i_{r+1}}}$ is disjoint from dependencies $D_{x_{j_{r+1}}},...,D_{x_{j_k}}$. That means

when we substitute $x_{i_{r+1}}$ for $x_{j_{r+1}}$, none of the clauses in $C^{\pm}(D_{x_{j_{r+2}}} \cup ... \cup D_{x_{j_k}})$ must be duplicated more often than in the original solution. As for the other clauses, we know that $dup_\Phi(\{x_{i_1}, ..., x_{i_{r+1}}\}) \leq dup_\Phi(\{x_{i_1}, ..., x_{i_r}, x_{j_{r+1}}\})$ due to the greedy selection criterion. It follows that
$dup_\Phi(\{x_{i_1}, ..., x_{i_{r+1}}, x_{j_{r+2}}, ..., x_{j_k}\}) \leq dup_\Phi(\{x_{i_1}, ..., x_{i_r}, x_{j_{r+1}}, ..., x_{j_k}\})$. □

So, as far as the overlapping of universal quantifier scopes is concerned, we obtain an optimal expansion schedule by greedily choosing the cheapest universal at a time. The theorem provides a good motivation for this greedy strategy, although it is clear that simply counting the sizes of all duplicated clauses is only a very rough upper bound for the actual costs of universal expansion and does not take into account the dynamics caused by simplifications. We now attempt to improve our heuristics by developing tighter cost estimates that can still be calculated efficiently enough for our preprocessing scenario.

5.5.2. Cost Estimation

In the discussion so far, our calculation of the expansion costs has come from the observation that expanding a universal x requires two copies of the existentials in D_x, one for $x = 0$ and one for $x = 1$, and accordingly also two copies of every clause that contains such an existential (Corollary 5.3.4). Using the notation from the last section, we have

$$c_x \leq |D_x| + \Sigma C^{\pm}(D_x)$$

where $\Sigma C^{\pm}(D_x)$ is the cumulative size of all clauses with a literal depending on x, and $|D_x|$ reflects the growth of the prefix (remember that the length of a QBF^* formula was defined to include the length of the prefix, so our cost estimate must also consider modifications to the prefix). To be precise, the prefix grows only by $|D_x| - 1$, as we can drop $\forall x$ after the expansion. This can be ignored here, because it creates the same constant offset for all universals in the formula.

According to the description in [Bie05], the scheduling of expansions and resolutions in the *QBF* solver Quantor does not take into account the locality of universals when calculating expansion costs. Instead of $\Sigma C^{\pm}(D_x)$ as in the estimate above, it uses $\Sigma C^{\pm}(Y_r)$ for the innermost existential quantifier block Y_r in the prefix. This overestimation matches the fact mentioned earlier that only universals from the innermost block X_r are taken into account in Quantor. Since our

preprocessor, on the other hand, does consider the whole prefix for the reasons given in Section 5.3.1, we need more precise dependencies D_x. Fortunately, in our preprocessing scenario, the number of scheduling decisions to be made is significantly lower than in a full solver like Quantor. One reason is that we do not have to schedule resolutions. Another reason is that the bound β_{exp} is typically so tight that we will only expand a limited number of universals, therefore we execute much less expansion cycles (iterations of the preprocessor's main loop). Accordingly, we can spend more time on selecting the variables and can afford to actually compute the sets of dependent existentials in each expansion cycle. This needs total time $O(e \cdot n \cdot |\Phi|)$ (cf. Listings 5.1 and 5.2), where e is the number of expansion cycles and n the number of universals in Φ. Our experiments show that this is still feasible: the total time spent for preprocessing is typically only a small fraction of the time required for the successive run of the solver. Furthermore, we assume that novel data structures might be applied here with great benefit in future versions of our preprocessor. We have already mentioned in Section 5.3.2 a very recent extension of our ideas by Lonsing and Biere [LB08b] with a simultaneous representation of dependency sets for all universals in a formula.

We adopt from Quantor a simple improvement of the above cost estimate by taking into account the obvious fact that when we let $x = 0$, all clauses that contain $\neg x$ can be removed, and we can drop all positive literals over x. Analogously, for $x = 1$, we drop all clauses with positive x and all literals $\neg x$:

$$c_x \leq |D_x| + \Sigma C^{\pm}(D_x) - \Sigma C^{\pm}(x) - |C^{\pm}(x)|$$

Expanding variables just because it is cheap to do so is a method without much foresight. We suggest to further improve our selection strategy by taking into consideration not only costs, but also goals which we might reach by expanding certain universals. A rewarding goal in solving satisfiability problems is to obtain unit literals. Propagating them helps keeping clauses short and might lead to discovering even more unit literals. This is in particular true for formulas with *2-CNF* subformulas, which might just collapse. Consider the following example:

$$\Phi = \forall x_1 \forall x_2 \exists y_1 \exists y_2 \ (x_1 \vee y_1) \wedge (\neg y_1 \vee y_2) \wedge (x_2 \vee \neg y_1 \vee \neg y_2)$$

The universals are pure variables, but we ignore this here for simplicity (perhaps, Φ is part of a larger formula). Then we have $D_{x_1} = D_{x_2} = \{y_1, y_2\}$, and

the expansion costs can be bounded as above by $c_{x_1} \leq 2+7-1-2=6$ and by $c_{x_2} \leq 2+7-1-3=5$, so we would select x_2 as the one with the lowest estimate. After expanding x_2 and simplifying the resulting formula by removing pure existential literals, we would get $\Phi' = \forall x_1 \exists y_1 \exists y_2\, (x_1 \vee y_1) \wedge (\neg y_1 \vee y_2) \wedge (\neg y_1 \vee \neg y_2)$ (there are no renamed existentials, because they were simplified away). Had we expanded x_1 instead, the whole matrix would have collapsed to the empty clause after propagating the unit literals y_1 and y_2 when $x = 0$. Of course, the same happens when we continue on Φ', because the possibility to obtain certain unit literals by expanding one universal is usually not lost by expanding a different one first. But since our preprocessing only expands a limited number of universals, we might be forced to stop well before discovering these units and might thus miss out on them.

Obtaining unit literals is not only a valuable goal to achieve, but as a simplification, unit propagation also has a direct influence on the costs of expanding a universal. That means we do not require separate measures for expansion costs and benefits, which would have to be balanced somehow. Instead, we can continue using costs as our single measure for choosing universals and simply subtract from the expansion costs the reductions that are achieved by propagating newly created unit literals.

For a universal variable x, let U_0 be the unit literals that would be obtained by a complete unit propagation under the assumption $x = 0$, and let U_1 be the units induced by $x = 1$. Then we can remove from the expanded formula all clauses in which a unit literal from U_0 or U_1 occurs, in addition to removing clauses containing $\neg x$ or x as above. With $\neg U_0 := \{\neg u \mid u \in U_0\}$ and analogously $\neg U_1$, we denote the set of negated occurrences of literals in U_0 or U_1. These can be dropped from all clauses which are not already deleted due to an occurrence of $\neg x$ or x. To avoid duplicate counting, we also have to take into account that occurrences of x and $\neg x$ for the cases $x = 0$ and $x = 1$ should only be deleted if they are in a clause without a unit literal. Then we obtain the following estimate:

$$\begin{aligned} c_x \leq\ & |D_x| + \Sigma C^{\pm}(D_x) \\ & - \Sigma C(U_0 \cup \{\neg x\}) - |C(\neg U_0) \setminus C(\neg x)| - |C(x) \setminus C(U_0)| \\ & - \Sigma C(U_1 \cup \{x\}) - |C(\neg U_1) \setminus C(x)| - |C(\neg x) \setminus C(U_1)| \end{aligned}$$

In order to compute this estimate, we have to perform for each universal variable x in each iteration of the preprocessor's main loop a complete unit propagation under the assumption that $x = 0$, and then under the assumption $x = 1$.

Since unit propagation can be done in linear time, this needs $O(e \cdot n \cdot |\Phi|)$ for e expansion cycles and n universals in Φ. As with the calculation of the variable dependencies above, we claim this is still feasible due to the relatively small values of e in our bounded expansion scenario.

There is an additional benefit to this computation: If U_0 (or analogously U_1) contains unit literals over existentials y_i whose quantifier precedes the quantifier of x, these literals must be unit literals for both cases $x = 0$ and $x = 1$, because they do not depend on x. That means we can propagate those units immediately (and remove them from U_0 or U_1), even without actually expanding x (similar to [Rin99]).

With the above estimate, we can take into account simplifications by unit propagation after one step of universal expansion. In a similar way, other kinds of one-step simplifications may be included. For example, subsumption can be considered by iterating over all clauses containing x or $\neg x$ and checking whether any such clause would subsume other clauses if $x = 0$ or $x = 1$, i.e. if the literal x or $\neg x$ is dropped from the given clause. Of course, this does not take into account that the propagation of units in U_0 and U_1 from above may also enable the subsumption of clauses. But considering such interdependence between different simplifications is likely too expensive to compute and can hardly be implemented non-destructively, so we only consider single simplifications in our cost estimate. Only after a particular universal has been chosen and has been expanded finally (i.e. destructively), circular application of simplifications until closure is performed as described in Section 5.4.

5.6. Integration of Q-Resolution

Besides expanding universal variables, it is also possible to eliminate existentials by performing all possible Q-resolutions on them as explained in Section 2.5, Proposition 2.5.2. We have already pointed out that this may lead to rapid formula growth, due to the potentially quadratic number of resolvents on each existential. Equally problematic is the hefty increase in the average clause length of a formula that typically comes with applying Q-resolution on a larger scale. Accordingly, our preprocessing focuses mainly on the expansion of universals. Nevertheless, a limited amount of resolution has proven helpful to counteract the

duplication of variables caused by universal expansion. More precisely, there are two situations in which we will apply resolution:

1. Whenever we can eliminate existentials without significantly increasing the formula size.
2. If we can use resolution specifically to reduce costs of a scheduled expansion.

In order to estimate the costs c_y of eliminating by Q-resolution an existential quantifier $\exists y$ from the innermost quantifier block, we use the upper bound given in [Bie05], adapted to our notation from the previous section:

$$c_y \leq |C(\neg y)| \cdot (\Sigma C(y) - |C(y)|) + |C(y)| \cdot (\Sigma C(\neg y) - |C(\neg y)|) - \Sigma C^{\pm}(y)$$

Here, the first part $r_y := |C(\neg y)| \cdot (\Sigma C(y) - |C(y)|) + |C(y)| \cdot (\Sigma C(\neg y) - |C(\neg y)|)$ represents the costs of generating all possible resolvents on y. For each occurrence of $\neg y$, that means $|C(\neg y)|$ times, the resolvents contain all literals in clauses over positive y ($\Sigma C(y)$), except y itself ($|C(y)|$), and analogously for all occurrences of y and clauses over $\neg y$. Finally, the term $-\Sigma C^{\pm}(y)$ in c_y models the fact that all original clauses over y and $\neg y$ are dropped from the formula after adding the resolvents.

The first occasion to apply resolution is during the simplification after expanding a universal. As long as there are existentials $\exists y_i$ from the innermost quantifier block that have estimated resolution costs below a threshold $c_{y_i} \leq \beta_{single\exists} \cdot |\Phi_{cur}|$, we continually pick the cheapest one and eliminate it by resolution. Here, $|\Phi_{cur}|$ is the current formula length after the last universal expansion, and the resolution bound $\beta_{single\exists}$ is a very small constant. We have achieved good results with values $\beta_{single\exists} \in [0, 0.003]$. With $\beta_{single\exists} = 0$, we would be guaranteed that the size of the formula does not increase.

Besides this general simplification that reduces the overall number of existentials, we also suggest a more specific application of resolution which only takes place after we have chosen a particular universal x for expansion. Our goal is to reduce its expansion costs c_x. A quick glance at the cost estimates from the last section shows that there are mainly two components which determine the value of c_x: the occurrences of $\pm x$ itself and the occurrences of dependent existentials. We are now going to apply resolution to attack the latter.

The idea is to resolve exclusively on dependent existentials in D_x immediately before expanding x. Eliminating such an existential $y \in D_x$ yields a double ben-

efit, because we do not only get rid of y itself, but also of its soon-to-be-created copy y'. In addition, we may also save copying some clauses during the following expansion. For example, a clause $(y_1 \vee y_2)$ with $y_1 \in D_x$ and $y_2 \notin D_x$ must be duplicated when x is expanded, but when we resolve on y_1 with $(\neg y_1 \vee y_3)$ and $y_3 \notin D_x$ before expanding x, the resolvent $(y_2 \vee y_3)$ does not need copying, since both literals do not depend on x. Of course, resolution usually produces many resolvents, some of which probably still require copying. In our example, the formula might also contain a clause $(\neg y_1 \vee y_4)$ with $y_4 \in D_x$, so that we obtain a second resolvent $(y_2 \vee y_4)$ which is still dependent on x.

Let δ_x be an estimate of the average fraction of clauses which must be duplicated when expanding x ($0 \leq \delta_x \leq 1$). Then we can estimate the costs $c_{y|x}$ of resolving an existential $y \in D_x$ before x is expanded:

$$c_{y|x} \;\approx\; (1+\delta_x) \cdot r_y \;-\; 2 \cdot \Sigma C^{\pm}(y)$$

We obtain this approximation from the upper bound for resolution given above. Again, $r_y := |C(\neg y)| \cdot (\Sigma C(y) - |C(y)|) \;+\; |C(y)| \cdot (\Sigma C(\neg y) - |C(\neg y)|)$ indicates the costs of performing all possible resolutions on y. The factor $(1+\delta_x)$ reflects the assumption that a portion δ_x of the resolvents is duplicated in the subsequent expansion of x. The original clauses over $\pm y$ would all have been copied when expanding x due to $y \in D_x$, hence the factor 2. For simplicity, we do not take into account that y and x might occur in common clauses. Since accurate values for δ_x would clearly be too expensive to compute, we use the estimate

$$\delta_x = 1 - \left(1 - \frac{|D_x|+1}{|vars(\Phi)|}\right)^l$$

if the formula contains a total number of $|vars(\Phi)|$ different variables and the average clause length is l. Then $(|D_x|+1)/|vars(\Phi)|$ is the probability that a randomly chosen variable is either x or in D_x. A clause of length l contains no such variable with probability

$$\left(1 - \frac{|D_x|+1}{|vars(\Phi)|}\right)^l .$$

Resolution also reveals an interesting special case. Consider a scenario in which we have a universal x with dependent existentials D_x where the set D_x can be

partitioned into subsets $D_x^{(1)} \cup D_x^{(2)} = D_x$ such that all $y_2 \in D_x^{(2)}$ are only connected via a single $y \in D_x^{(1)}$ to x or other existentials $y_1 \in D_x^{(1)}$. That means for all $y_2 \in D_x^{(2)}$, we have $y_2 \not\sim x$ and $y_2 \not\sim y_1$ for all $y_1 \in D_x^{(1)}$ with $y_1 \neq y$. Figure 5.2 illustrates the situation.

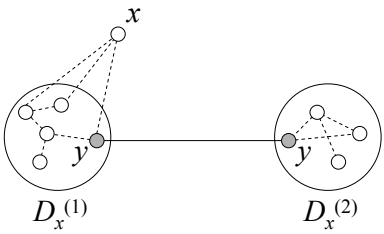

Figure 5.2.: Dependencies with a single link

So, y is the only link which propagates dependency from $\{x\} \cup D_x^{(1)}$ on the one hand to $D_x^{(2)}$ on the other hand. Can we destroy that link to make the existentials in $D_x^{(2)}$ independent from x? We could eliminate y by resolving on it, but for non-tautological clauses of the form $(y^\varepsilon \vee y_1^{\varepsilon_1} \vee \alpha)$ and $(y^{1-\varepsilon} \vee y_2^{\varepsilon_2} \vee \beta)$, this creates resolvents $(y_1^{\varepsilon_1} \vee y_2^{\varepsilon_2} \vee \alpha \vee \beta)$. Then $y_1 \sim y_2$ for $y_1 \in D_x^{(1)}$ and $y_2 \in D_x^{(2)}$, which means Q-resolution usually creates direct links between $D_x^{(1)}$ and $D_x^{(2)}$. However, this does not happen if we only have clauses $(y^\varepsilon \vee y_1^{\varepsilon_1} \vee \alpha)$ and $(y^\varepsilon \vee y_2^{\varepsilon_2} \vee \beta)$ which themselves cannot be resolved on y, because it occurs in the same polarity y^ε, and all occurrences of $y^{1-\varepsilon}$ are in clauses that contain neither x nor any existential in D_x. Then we can resolve away y, and the existentials in $D_x^{(1)}$ and $D_x^{(2)}$ will not be connected anymore, since y has been replaced with variables that do not propagate the dependency. The situation is still the same when there are multiple such existentials that connect $D_x^{(1)}$ and $D_x^{(2)}$ with only one polarity each. We can eliminate them one after another and unlink $D_x^{(2)}$ from x and $D_x^{(1)}$.

In this scenario, the special property is that for a universal x_j, we have an existential $y \in D_{x_j}$ for which one polarity of y occurs only together with other variables v with $v \notin D_{x_j}$ and $v \neq x_j$. A closer investigation reveals that we do not need to perform the actual resolution. Instead, we can immediately infer that y does in fact not depend on x_j.

Theorem 5.6.1. *(Reduction of One-Sided Dependencies)*
Given $\Phi \in QBF^*$, let $\forall x_j$ be a universal quantifier in Φ, and let $\exists y_i$ be an existential in the scope of $\forall x_j$, such that one polarity of y_i only occurs in clauses over y_i and other variables v with $v \notin D_{x_j}$ and $v \neq x_j$. Then Φ has an equivalence model $M = (f_{y_1}, ..., f_{y_m})$ in which f_{y_i} does not depend on x_j.

Proof:
Assume that Φ has no equivalence model in which f_{y_i} does not depend on x_j. Let M' be an equivalence model in which y_i is mapped to the model function $f'_{y_i}(z_1, ..., z_r, x_1, ..., x_{j-1}, x_j, x_{j+1}, ..., x_k)$ that needs to depend on x_j. Then there exists an assignment t of truth values to all free and universal variables such that

$$\begin{aligned} & f'_{y_i}(t(\mathbf{z}), t(x_1), ..., t(x_{j-1}), 0, t(x_{j+1}), ..., t(x_k)) \\ \neq\ & f'_{y_i}(t(\mathbf{z}), t(x_1), ..., t(x_{j-1}), 1, t(x_{j+1}), ..., t(x_k)) \end{aligned}$$

and the formula matrix is true for $x_j = 0$ and $x_j = 1$ if the values from t are substituted for the free and the other universal variables and the existentials are assigned as determined by the equivalence model. Let τ be the value of the expression in the first line. Without loss of generality, assume that $\neg y_i$ is the polarity of y_i which occurs only in clauses with other variables $v \notin D_{x_j}$ and $v \neq x_j$. Then none of those clauses contains a variable which depends on x_j, yet those clauses remain true when y_i flips from τ to $\neg\tau$ as x_j changes. That means those clauses are true regardless of the value of y_i. Thus, we can choose the same truth value $y_i = 1$ for both $x_j = 0$ and $x_j = 1$, and all clauses with positive y_i will be satisfied as well. It follows that M' remains an equivalence model if we let

$$f'_{y_i}(t(\mathbf{z}), t(x_1), ..., t(x_{j-1}), x_j, t(x_{j+1}), ..., t(x_k)) := 1 \ .$$

By using this argument repeatedly, we obtain an equivalence model in which f'_{y_i} does not depend on x_j, which is a contradiction to our initial assumption. □

It is also possible to apply the theorem consecutively. Assume that it holds for some y_i and that we have another existential y_l that occurs in one polarity only in clauses over variables v with $v = y_i$ or $v \notin D_{x_j} \cup \{x_j\}$. Then y_l also does not depend on x_j. This can be seen by replacing y_i with an equivalence model function f_{y_i} that does not depend on x_j. After retransformation of the resulting formula into *CNF*, one polarity of y_l shares only clauses with variables $v \notin D_{x_j}$ and $v \neq x_j$, so y_l also satisfies the requirements of the theorem. We can actually remove both y_i and y_l from D_{x_j}.

This result can be combined with the dependency computation techniques from Section 5.3. We can, for example, use a linear-time marking algorithm to compute first without considering variable polarity all existential dependencies of a given universal. In a second step, we then iterate over all marked clauses and determine whether a clause contains only one dependent (marked) existential. If this is the case, we increase a counter for that variable and its corresponding polarity. In a suitable *CNF* data structure, this can be done in time linear in the formula length. Then we iterate over all existential variables and check whether there is a variable for which the total number of occurrences in one polarity equals the previously computed counter value. If this is the case, we can apply the above theorem and remove the variable from the dependent existentials. Then we look through all clauses in which this variable occurs to find new clauses that now have only one dependent existential, and so on. In total, this requires time $O(m \cdot |\Phi|)$ if there are m existential quantifiers in Φ. To compute the dependencies for all universals in the formula in this way, we thus need cubic time, compared to the quadratic complexity of the original marking algorithm without variable polarity. Of course, we can attempt to improve the performance of the algorithm by taking into account polarity already during the initial marking, but this does not appear to lower the actual complexity. It is currently an actively investigated question in the *QBF* research community whether there exists a quadratic algorithm for computing such dependencies with variable polarity.

On the other hand, the consideration of variable polarity can significantly reduce the amount of copying involved in universal expansion and can thus easily offset the additional complexity in computing the dependent existentials. It is not difficult to see that in the bast case, the inclusion of variable polarity can completely eliminate all dependencies D_x of a universal x and thus give an arbitrarily large improvement. A simple example is the following formula:

$$\forall x \exists y_1 \ldots \exists y_m \, (z_1 \vee \neg y_1) \wedge \ldots \wedge (z_{m-1} \vee \neg y_{m-1}) \wedge (x \vee y_m) \wedge (y_1 \vee \ldots \vee y_{m-1} \vee \neg y_m)$$

Without considering variable polarity, we have $D_x = \{y_1, \ldots, y_m\}$. But with polarity, this reduces to $D_x = \emptyset$, so it is actually sufficient to perform only universal reduction to eliminate the occurrence of x in the last clause.

As mentioned in Section 5.2, a very similar result has been found independently and simultaneously with our initial publication [BKB07] by Samer and Szeider in [SS07]. The difference is essentially in the notation, where they also consider

universals as being dependent of enclosing existentials. This might be useful for QBF^* solvers based on other techniques besides universal expansion or resolution, but it also makes the notation more complex and less intuitive. In any case, we consider this simultaneous result on variable dependencies as a confirmation of the importance that this topic appears to have for QBF^* reasoning.

5.7. Implementation and Experiments

All the techniques described in this chapter have been implemented into an expansion-based preprocessor for QBF^* formulas in clausal form. Our implementation has been designed with two main objectives in mind:

1. to provide an experimental verification of the contributions made in this chapter. These can be summarized into two main aspects: we have suggested bounded universal expansion for preprocessing, and we have developed various refinements to universal expansion. That means we want our experiments to show on the one hand that solvers based on other techniques can indeed benefit from preprocessing by bounded universal expansion. On the other hand, we want to demonstrate that our refinements of universal expansion can increase the performance of expansion itself.
2. to build a platform that facilitates future experiments on universal expansion and Q-resolution, and also investigations on models for quantified Boolean formulas. This work has already been put to use in Section 3.4 to study random *QHORN* formulas.

5.7.1. A Software Platform for QBF^*

In line with the second objective, we have decided to focus on extensibility and reusability, rather than on performance optimizations. Accordingly, the preprocessor has been implemented on top of our existing Java-based logic framework ProverBox [Bub03]. The primary goal of the framework is to integrate different logics and different theorem proving algorithms. The initial version has supported propositional and predicate logic, and we have now added QBF^*.

Inevitably, the genericity of such a multi-logic framework does come with a performance overhead. For example, QBF^* solvers usually represent literals as

signed integers, but in the ProverBox framework, literals are objects with an explicit Boolean flag for their polarity and a reference to an object that implements the `Atom` interface. An `Atom` then contains an `AtomSymbol` whose name is a `String`. This allows *CNF* data structures and algorithms on them to be shared between the various logics, but it is clear that operations like checking for equality or complementarity of two literals are significantly more expensive in this general framework.

Despite these performance concerns, we can already reveal that in the experiments below, the preprocessing time for most benchmark families is rather small compared to the time required by the subsequently called solvers. So it appears that the performance of the current preprocessor implementation is adequate for our experiments. For maximum performance, we would recommend our expansion techniques to be integrated directly into a competitive solver and to use more sophisticated data structures for computing and storing dependencies, perhaps based on the previously mentioned idea from [LB08b].

On the other hand, our generic framework has allowed us to quickly build a powerful preprocessor with a clean architecture and lots of additional features for experimenting with *QBF**:

- The software does not only provide a fully automatic preprocessor, but it can also be used interactively. The user can then selectively expand arbitrary universal quantifiers, apply formula transformations, view formula statistics, compute Horn renamings, etc.
- In addition to reading and writing files in the standard DIMACS and QDIMACS formats, a formula parser also allows inputs in familiar formula notation for all supported logics.
- A generator for random *QBF* and *QHORN* formulas (see Section 3.4.1) is included.
- 3rd party modules can implement additional functionality via a plugin interface. It provides easy access to the ProverBox data structures and the included algorithms, such as universal expansion, Q-resolution and all the simplifications from Section 2.4.

Figure 5.3 provides a screenshot of the ProverBox main window. The large frame on the left is the integrated editor showing a part of the QBFLIB formula *Adder2-4-c*. In the command prompt at the bottom, the preprocessor has been

invoked, and its output is shown in the right-hand frame. We can see detailed statistics about the effect of the preprocessing on the given formula.

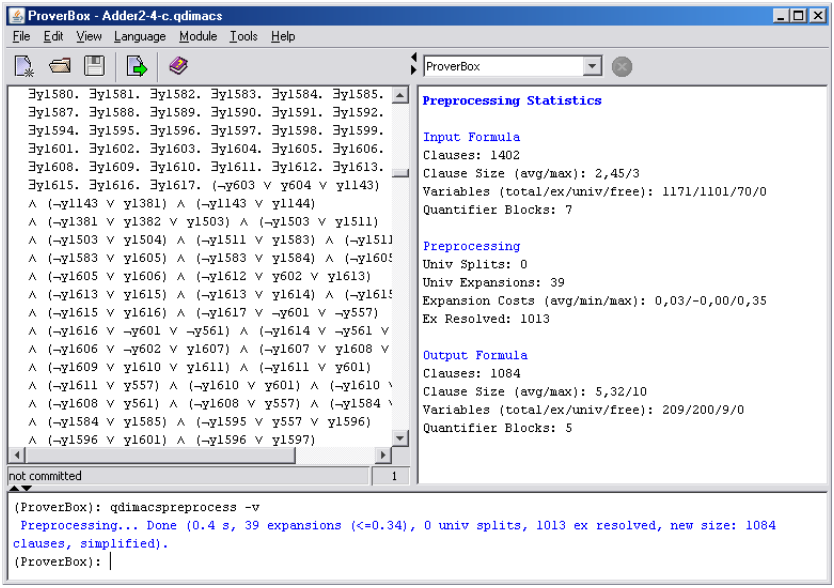

Figure 5.3.: Screenshot of the ProverBox main window

The complete ProverBox software for propositional calculus, predicate logic and *QBF** is available from `www.ub-net.de/cms/proverbox.html`.

5.7.2. Experiment Setup

We have conducted our experiments with three state-of-the-art *QBF* solvers, each in its latest publicly available version[2]: Quantor 3.0 [Bie05], sKizzo 0.8.2 [Ben05a] and SQBF 12/06 [SB05]. These solvers are well known and have each made it into the top 3 places (YASM rating for non-probabilistic formulas) at least once in the 2006 - 2008 competitions of QBFEVAL, a series of international competitive evaluations of *QBF* solvers (`www.qbfeval.org`).

[2]as of August 2009.

Quantor, sKizzo and SQBF are all based on different techniques, so we can examine how useful our preprocessing is for different solving methods. Since Quantor is built on the same general idea of universal expansion as our preprocessor, potential improvements will mainly come from our expansion refinements, and not so much from the preprocessing itself. That means the comparison with Quantor is a good way to evaluate the effectiveness of these refinements. sKizzo mainly uses symbolic skolemization by successively computing satisfiability model functions, and SQBF is an improved variant of the DPLL algorithm for *QBF* that attempts to solve *QBF* formulas by working on their propositional matrix with a conventional SAT solver for different assignments to quantified variables. In both cases, preprocessing by universal expansion should be very useful: for sKizzo, the arity of the model functions should decrease, and SQBF should need fewer decisions on universal variables, and thus fewer calls to the SAT solver.

The experiments have been run on two platforms: Cygwin 1.5 on Windows Vista x64 on a machine with Core2Duo E8400 at 3.6 GHz and 4GB RAM for Quantor and SQBF, and Debian Linux 5.0 64bit on a Core2Duo E6600 with 2.4 GHz and 4GB RAM for sKizzo. We would have preferred to use the same platform for all three solvers, but we had found the Cygwin binary for sKizzo to be unstable on our Vista x64 machine. Nevertheless, working with two different platforms does not have a negative impact on the expressiveness of our experiments, since our main objective is not to compare the different solvers with each other. Instead, we want to compare for each single solver the results with and without preprocessor. Of course, the preprocessor has been run on the same machine as the corresponding solver in each experiment. Both machines have been equipped with Java 1.6 for this purpose. The solvers and our current preprocessor implementation all use only one single working thread and thus only marginally benefit from a dual core processor.

Each solver has been run with default parameters on 15 families of benchmarks with a total of 849 formula instances from the QBFLIB collection [GNT01]. We have tried to choose benchmark families of such a difficulty level that the solvers could solve some, but not all, formulas of a family within a time limit for each formula of 300 seconds. Two attempts have been made to solve a formula: one with the solver itself, and one where the formula is first fed into our preprocessor and its output is then given to the solver (unless the preprocessor has already solved the formula on its own). Of course, the time limit and the time recorded for the experiments with preprocessor consider both the preprocessor and the

solver in combination. That means if the preprocessor requires 50 seconds on a formula, the solver has 250 seconds left until timeout.

It may happen that the preprocessor itself does not finish within 300 seconds, but we will see in the detailed results below that this has occurred in our experiments only with 2-3 very large instances on which the solvers alone have mostly timed out as well. In these rare cases, it might have been helpful to restrict the preprocessing to a fixed fraction of the given time, e.g. 150 seconds, and hope that some partial preprocessing might be sufficient for the solver to handle the formula. We did, however, refrain from such ideas, because preprocessing times are in most cases only a small fraction of 300 seconds, which makes special treatment for only a few formulas somewhat pointless. In addition, internal time limitations are explicitly forbidden by the rules of the QBFEVAL, which we have taken as a role model for our own experiments. As usual, if an instance cannot be solved, e.g. because of a timeout or an out-of-memory condition, it is counted as the timeout value.

To make the experiments less time-consuming, we have performed them in a give-up mode in which a (sub)family of formulas is quit whenever we encounter an instance that is unsolved by both the solver and the solver with preprocessor. For example, neither sKizzo itself nor sKizzo with preprocessor can solve the instance *adder-14-sat*, so we skip the next and also last instance of the same kind, *adder-16-sat*, without counting this one as a timeout, and continue immediately with *adder-2-unsat*. Of course, this requires that the instances are approximately sorted in order of ascending difficulty. Where this was not already the case by default, we have grouped formulas in obvious subfamilies (e.g. *adder-sat*, *adder-unsat*, *Adder2-s*, *Adder2-c*, or *cnt*, *cnt-r*, ...).

It is clear that the expansion bound β_{exp} has a significant impact on the performance of our preprocessing approach. We will later see that the solvers react differently to modifications in this parameter, so there is not one single value which is optimal for all solvers. We have, however, found that $\beta_{exp} = 2.0$ (in combination with a resolution bound of $\beta_{single\exists} = 0.002$) is a good compromise. And since our goal is to provide a multi-purpose preprocessor that does not require parameter tweaking by the user, we have decided to conduct most of the following experiments with these default values, rather than using individual optimal bounds for each solver. A more detailed comparison of parameter settings will then be given in Section 5.7.4.

5.7.3. Results and Discussion

An overview of our experimental results can be seen in Table 5.1. It presents for each solver and each combination of a solver with our preprocessor the total number of instances solved and the total computation time in seconds. For better comparisons, we have also included the relative improvements. In this and in the following tables, we use bold font to indicate best results (within a category). The results given here are a summary of Tables 5.2 - 5.4 on pages 168 - 170, which will be discussed further below.

Table 5.1.: Summary of preprocessing results
(default bounds $\beta_{exp} = 2.0$, $\beta_{single\exists} = 0.002$)

	Quantor	*Quantor+pre*	*sKizzo*	*sKizzo+pre*	*SQBF*	*SQBF+pre*
#Solved	194	**215**	300	**331**	212	**293**
Increase		**+10.8%**		**+10.3%**		**+38.2%**
Time [sec.]	16,345	**10,854**	21,416	**10,876**	35,589	**10,590**
Speedup		**≥1.5**		**≥2.0**		**≥3.4**

All three solvers show significant overall gains from the preprocessing. In particular, our preprocessor allows SQBF to solve almost 40% more problems in less than 30% of the time originally recorded. If we consider only the computation time, that is a more than three-fold speedup. But it is likely that some of the problems which SQBF alone cannot solve within 300 seconds require far more time than that in order to be solved by SQBF without preprocessor. That means the real speedup might be much higher, perhaps an order of magnitude or even more on some problems. We have tried to emphasize this with the $\geq$ signs in front of the speedup values in the table. On many of the problems solved by SQBF with preprocessor, SQBF alone appears to run out of memory (it heavily uses clause learning, which might be the reason for that memory consumption), so the advantage from preprocessing cannot even be offset with more computation time.

The reason why SQBF benefits so much from universal expansion appears to be the heavy influence that the number of universal quantifiers and the number of quantifier blocks has on DPLL-style solvers. If there were only existential

quantifiers in the input formula, SQBF would require only one run of the SAT solver on the matrix of the formula, but this number might grow exponentially for more universals. For the skolemization-based solver sKizzo, the size of the model functions may also grow exponentially in the number of universals, but we assume that the benefits from having fewer universals are partially absorbed by the increase in the number of existentials that may come with universal expansion. That means sKizzo might need to find more model functions, and that focus on the existentials is probably the main reason why the preprocessing effect is not as strong as with SQBF.

The good results with Quantor are somewhat surprising, considering that Quantor is also expansion-based and therefore cannot be expected to benefit that much from the preprocessing itself, but only from our refinements of universal expansion. It appears that our improved dependency scheme, the careful selection of universals from the whole prefix and the resolution on dependent existentials can indeed considerably reduce the memory consumption of repeated universal expansion. In many of the problems that are only solved with the help of the preprocessor, Quantor on its own seems to run out of memory, just like we have observed with SQBF.

We now consider the more detailed itemization of results for individual benchmark families given in Tables 5.2 - 5.4 on pages 168 - 170. All time values are wall time measured in seconds. Time values shown in the 6th column are the total times for running solver and preprocessor in combination on all attempted problems in the corresponding benchmark family. In parentheses, the fraction of that time spent on preprocessing is shown (rounded to whole percentages). For example, the total computation time for Quantor with preprocessor on the Adder family was 1,218 seconds, of which about 1% (the exact value is 17 seconds) were for preprocessing.

We can observe that preprocessing times are negligible for most families, but there are a few families where practically all computation time is spent on preprocessing. Some of the latter are completely solved by the preprocessor alone without calling the solver at all (ASP, Szymanski), or they are dramatically simplified by the preprocessing (e.g. Sakallah s499, s510). The number of solved problems increases in almost all these benchmark families that require a significant amount of preprocessing time, so the time appears well spent.

Table 5.2.: Benchmark results for Quantor without/with preprocessing
(default bounds $\beta_{exp} = 2.0$, $\beta_{single\exists} = 0.002$; times in seconds)

Benchmark Family		*Quantor*		*Quantor + preproc*			*Speedup*
Name	*#inst*	*#solvd*	*time*	*#solvd*	*time (preproc)*		
Adder	32	8	1,812	**10**	**1,218**	(1%)	**1.5**
ASP	12	0	3,600	**12**	**95**	(100%)	**37.9**
Blackbox_design *.003	8	0	300	0	300	(0%)	1.0
Blocks	13	13	**89**	13	113	(2%)	*0.8*
Connect3 cf_3_3*	21	4	302	4	302	(0%)	1.0
Counter	88	54	3,184	**56**	**2,744**	(0%)	**1.2**
CounterFactual 4; 8-16	480	49	2,708	**50**	**2,404**	(0%)	**1.1**
Evader-Pursuer 4x4-lg	7	1	336	1	**334**	(0%)	1.0
k_branch_n	21	3	302	3	302	(0%)	1.0
k_path_n	21	21	**2**	21	7	(71%)	*0.3*
RobotsD2 *.2, *.4, *.8	29	9	**612**	9	646	(1%)	*0.9*
Sakallah s499, s510	13	3	928	**4**	**743**	(100%)	**1.2**
Sorting_networks	84	26	660	26	**659**	(0%)	1.0
Szymanski	12	3	910	**5**	**525**	(100%)	**1.7**
Term1	8	0	600	**1**	**462**	(0%)	**1.3**
Total	849	194	16,345	**215**	**10,854**	(13%)	**1.5**

Table 5.3.: Benchmark results for sKizzo without/with preprocessing (default bounds $\beta_{exp} = 2.0$, $\beta_{single\exists} = 0.002$; times in seconds)

Benchmark Family		*sKizzo*		*sKizzo + preproc*			*Speedup*
Name	*#inst*	*#solvd*	*time*	*#solvd*	*time (preproc)*		
Adder	32	10	1,536	**11**	**1,381**	(4%)	**1.1**
ASP	12	8	1,799	**12**	**147**	(100%)	**12.2**
Blackbox_design *.003	8	0	300	0	300	(0%)	1.0
Blocks	13	8	300	8	300	(0%)	1.0
Connect3 cf_3_3*	21	1	300	1	300	(0%)	1.0
Counter	88	49	4,041	**54**	**2,564**	(0%)	**1.6**
CounterFactual 4; 8-16	480	171	3,378	**174**	**1,940**	(1%)	**1.7**
Evader-Pursuer 4x4-lg	7	1	**315**	1	328	(0%)	1.0
k_branch_n	21	5	649	**6**	**307**	(1%)	**2.1**
k_path_n	21	10	1,574	**14**	**842**	(0%)	**1.9**
RobotsD2 *.2, *.4, *.8	29	18	4,705	**29**	**96**	(88%)	**49.0**
Sakallah s499, s510	13	2	1,201	**4**	**812**	(100%)	**1.5**
Sorting_networks	84	12	601	12	601	(0%)	1.0
Szymanski	12	5	**417**	5	658	(100%)	*0.6*
Term1	8	0	300	0	300	(0%)	1.0
Total	849	300	21,416	**331**	**10,876**	(16%)	**2.0**

Table 5.4.: Benchmark results for SQBF without/with preprocessing
(default bounds $\beta_{exp} = 2.0$, $\beta_{single\exists} = 0.002$; times in seconds)

Benchmark Family		*SQBF*		*SQBF + preproc*			*Speedup*
Name	*#inst*	*#solvd*	*time*	*#solvd*	*time (preproc)*		
Adder	32	4	2,103	**7**	**1,233**	(1%)	**1.7**
ASP	12	0	3,600	**12**	**91**	(100%)	**39.6**
Blackbox_design *.003	8	0	300	0	300	(0%)	1.0
Blocks	13	11	610	**12**	**400**	(1%)	**1.5**
Connect3 cf_3_3*	21	14	2,144	**21**	**317**	(3%)	**6.8**
Counter	88	37	4,164	**42**	**2,965**	(0%)	**1.4**
CounterFactual 4; 8-16	480	100	11,580	**126**	**1,553**	(0%)	**7.5**
Evader-Pursuer 4x4-lg	7	7	**9**	7	19	(26%)	*0.5*
k_branch_n	21	4	1,599	**8**	**502**	(0%)	**3.2**
k_path_n	21	5	1,316	**8**	**559**	(0%)	**2.4**
RobotsD2 *.2, *.4, *.8	29	20	2,810	**29**	**124**	(44%)	**22.7**
Sakallah s499, s510	13	0	1,800	**4**	**746**	(100%)	**2.4**
Sorting_networks	84	10	1,154	**11**	**659**	(0%)	**1.8**
Szymanski	12	0	1,800	**5**	**525**	(100%)	**3.4**
Term1	8	0	600	**1**	**597**	(0%)	1.0
Total	849	212	35,589	**293**	**10,590**	(14%)	**3.4**

In each of Tables 5.2 through 5.4, at least half the benchmark families show an increase in the number of solved problems when using our preprocessor. For SQBF, almost all families (13 out of 15) benefit, and every solver has families where more than 50% additional problems can be solved with preprocessing. Also very important is the fact that in our experiments, the number of solved problems has never decreased due to the preprocessing, and that applies to all three solvers. Only in 6 out of $3 \cdot 15$ cases, a solver was faster on a family without preprocessor. We think this supports our initial hypothesis from Section 5.4 that enforcing tight bounds on the expansion can largely prevent negative effects. In total, we think it is safe to say that the positive effects of our preprocessing by far outweigh its potential drawbacks.

A close look at the benchmark families in Tables 5.2 through 5.4 reveals that there are some families for which every solver significantly benefits from the preprocessing, e.g. Adder, ASP or Sakallah s499/s510. On the other hand, no solver seems to take advantage of the preprocessing on the two benchmark families Blackbox_design and Evader-Pursuer. The Blackbox formulas are probably just too hard, because none of the solvers in our experiment is able to solve any such instance, even if the timeout is increased to much more than 300 seconds. We have decided to keep this family in our experiments nonetheless to provide at least one example for such a very hard formula class. To understand the results for Evader-Pursuer, the other family that does not show any benefits from the preprocessing, we need to consider in more detail the structure of these formulas, and how the preprocessing affects it.

Table 5.5 shows for each family the average impact that the preprocessing has on various formula metrics. The first two columns of data indicate the average relative difference in the number of universal and existential quantifiers before and after preprocessing. For example, assume that a family consists of two formulas, one with 10 universal quantifiers before preprocessing and 5 of them after preprocessing, and another one with 1 universals before and 0 universals after preprocessing. Then we have -50% universals for the first formula and -100% for the second one, which results in an average relative difference of -75%. By first calculating relative differences and then taking the average, we give the same weight to every formula. The third column of data reports the average relative differences in the number of quantifier blocks in the prefix, and the last two columns provide average relative differences in the number of clauses and in the average number of literals per clause. The last line in the table contains the averages of all entries in the corresponding column.

Table 5.5.: Effects of preprocessing on the formula structure (default bounds $\beta_{exp} = 2.0$, $\beta_{single\exists} = 0.002$; shown are average relative differences, where negative values indicate an improvement by preprocessing)

Benchmark Family	*Universals*	*Existentials*	*Prefix blocks*	*Clauses*	*Ø Clause size*
Adder	**-62.2%**	**-43.4%**	**-15.5%**	+12.1%	+104.9%
ASP	**-100.0%**	**-100.0%**	**-100.0%**	**-100.0%**	**-100.0%**
Blackbox_design *.003	**-32.0%**	**-53.2%**	**-17.9%**	+10.3%	+45.6%
Blocks	**-20.9%**	+12.1%	0.0%	+9.7%	+0.4%
Connect3 cf_3_3*	**-72.8%**	**-70.8%**	**-72.8%**	**-60.1%**	**-64.6%**
Counter	**-58.8%**	**-6.9%**	**-54.7%**	+20.1%	**-6.9%**
CounterFactual 4; 8-16	**-39.4%**	**-19.3%**	**-20.5%**	**-0.1%**	+38.2%
Evader-Pursuer 4x4-lg	**-4.1%**	+14.3%	0.0%	+60.2%	+6.8%
k_branch_n	**-15.5%**	**-4.1%**	**-14.8%**	+58.1%	+12.6%
k_path_n	**-36.4%**	+31.4%	**-33.9%**	+28.6%	+14.8%
RobotsD2 *.2, *.4, *.8	**-5.7%**	+64.7%	0.0%	+29.7%	+2.0%
Sakallah s499, s510	**-83.5%**	**-87.4%**	**-60.0%**	**-87.5%**	**-23.5%**
Sorting_networks	**-23.4%**	**-27.5%**	0.0%	+13.6%	+35.5%
Szymanski	**-100.0%**	**-100.0%**	**-100.0%**	**-100.0%**	**-100.0%**
Term1	**-54.8%**	+1.0%	**-37.7%**	+44.7%	+22.9%
Family Average	**-47.3%**	**-25.9%**	**-35.2%**	**-4.0%**	**-0.8%**

If the preprocessor manages to completely solve a formula instance, we count −100% for all metrics of this formula. We have used bold font for all values which are negative and thus indicate an improvement by preprocessing. The table represents all of the 849 benchmark instances that could be preprocessed within the time limit, which excludes the 7 largest Szymanski instances and the Sakallah formulas s499_d8_s / s510_d8_s and upwards. None of those could be solved by any of the solvers within the same time limit.

We can first observe that on average over all families, the number of universals is almost cut in half, and the number of quantifier blocks in the prefix is reduced by more than one third. Independent of the speedups and increases in the number of solved problems that we have reported earlier, these new numbers are another demonstration of the effectiveness of bounded universal expansion for preprocessing. At the same time, the number of clauses and the average clause size remain roughly the same, with the existentials actually decreasing as well. This suggests that the strategy of resolving primarily on dependent existentials prior to a scheduled universal expansion does indeed help in keeping the formulas small.

A nice example in absolute numbers is the Adder2-s subfamily where the mean number of universals before preprocessing is 557, and 110 afterwards. That means more than 440 universals can be eliminated on average with at most a doubling of the formula size. Considering that the naive expansion of universal quantifiers would almost double the formula size for each single universal, this is certainly an impressive result. At this point, we would like to mention that those Adder2-s formulas are not particularly easy to solve: the larger instances have been classified as “HARD” in the latest QBFEVAL, and in our experiments, none of the solvers has succeeded on more than half of them. Even in those cases in which the preprocessor is able to eliminate all universals (which is possible within the given bounds for the two smallest instances), the resulting propositional formula does not collapse and cannot be solved by preprocessing alone.

Adder is one of the families where preprocessing helps all three solvers. The other families in this category are ASP, Counter, CounterFactual and Sakallah. Table 5.5 shows that all these families have relatively large reductions in the number of universals and also in the number of quantifier alternations. Since the expansion bound β_{exp} is always the same, these families obviously have universals that are cheaper to expand than the universals in other benchmark

families. The observation that families with cheap universals tend to show the highest speedups over the original solvers seems to verify our hypothesis that universal expansion is most powerful for preprocessing when it is restricted to the cheapest universals, whereas more expensive universals should be attacked with different techniques. Of course, this is only a rough characterization. According to Table 5.5, Connect3 and Szymanski also seem to behave very well under universal expansion. As expected, we can see some nice speedups for these families, but only with one or two solvers, not with all three. The reason is probably that in both families, many problems can be solved by the preprocessor alone, which has a positive impact on the formula metrics for those families, but hides the effort for the preprocessing and overshadows the remaining formulas (in Connect3) for which universal expansion is more expensive.

A family in which none of the solvers benefits from preprocessing is Evader-Pursuer. In these formulas, we have found all universals to be very expensive to expand. We can see from Table 5.5 that on average, only 4.1% of the universal quantifiers in an Evader-Pursuer instance can be expanded. In absolute numbers, that is one universal per formula, but that single expansion causes a 60% increase in the number of clauses (and that already includes subsequent simplifications), which is the largest increase of all families in the experiment. Evader-Pursuer encodes a chess-like game where the universal quantifiers indicate the moves of one of the two players. Apparently, every single move heavily influences the valuation of the whole game. Clearly, it would be unwise to expand more universals in that situation. Indeed, if we loosen the expansion bound to allow two expansions with Evader-Pursuer, both sKizzo and SQBF will perform worse on this family. Do the families with cheaper universals show the same behavior when the bound is increased? We will try to answer this question in the next section.

5.7.4. Comparison of Different Expansion Bounds

We would like to finish our experiments with a brief comparison of different values for the expansion bound β_{exp}. This is the parameter which directly controls the number of expansions performed and therefore determines the performance of the whole preprocessor. The second parameter, the resolution bound $\beta_{single\exists}$, has a much more indirect influence, since our preprocessing approach uses resolution only as a simplification to mitigate the increase of existential variables

due to universal expansions. A similar threshold for unscheduled resolutions has already been investigated in [Bie05]. Biere suggests a fixed threshold of 50, while we have decided to choose a relative bound of $\beta_{single\exists} = 0.002 \cdot |\Phi|$ for a formula of length $|\Phi|$. For many small to medium-sized problems in the above benchmark families, $0.002 \cdot |\Phi|$ is in the range of 20 to 60 and therefore roughly of the same order of magnitude as the value suggested by Biere, so we focus in the following on the more important parameter β_{exp}.

Figure 5.4 shows for different expansion bounds $\beta_{exp} \in [1.0, 4.0]$ how effective the preprocessor is in terms of solved problems in combination with SQBF. For each bound, the diagram indicates the relative increase in solved instances that SQBF with β_{exp}-bounded universal expansion can achieve in comparison to SQBF without preprocessor.

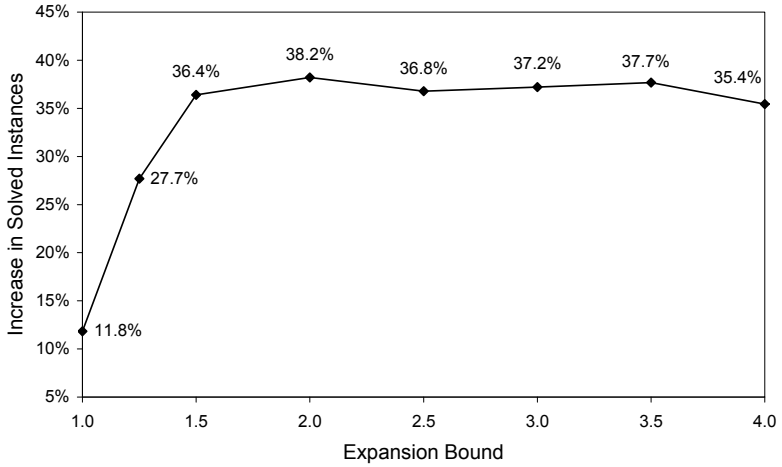

Figure 5.4.: SQBF preprocessing benefit for different bounds β_{exp} (default resolution bound $\beta_{single\exists} = 0.002$)

At $\beta_{exp} = 1.0$, only the general simplifications described in Section 5.4 are applied. Obviously, SQBF has less powerful simplifications, since there is already an improvement even without universal expansion, but the effect is still small in comparison to the peak at approximately $\beta_{exp} = 2.0$, the default bound from

the previous experiments. sKizzo and Quantor are capable of essentially the same general simplifications as our preprocessor, so they hardly benefit from preprocessing without expansions, as can be seen in the following Figure 5.5 for Quantor.

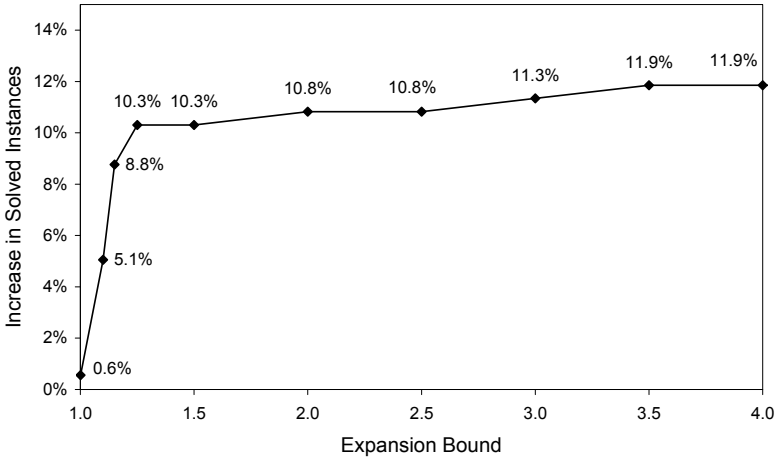

Figure 5.5.: Quantor preprocessing benefit for different bounds β_{exp} (default resolution bound $\beta_{single\exists} = 0.002$)

The most interesting difference between the two figures is the following: for larger bounds, the preprocessor performance with SQBF seems to slowly decline, whereas it appears to further increase with Quantor.

The behavior with SQBF is what we would expect from our previously mentioned assumption that cheap universals should be expanded, and expensive ones should be left as they are, because the subsequent solver might be better at handling them with a different technique. If the expansion bound β_{exp} is increased, it means that the preprocessing extends to more expensive universals. Figure 5.4 shows that SQBF seems indeed better at dealing with these expensive universals, but only slightly, since the curve appears to fall rather slowly. It might also be the case that with larger bounds, the sheer size of the expanded formula is becoming a problem for SQBF. We have to consider that many problems with

only a few thousand clauses are already considered hard in the QBFLIB. Now if the input formula is very large with, say, 100,000 clauses, a four-fold growth might really cause a solver to run into problems like memory exhaustion or too expensive operations, even if the formula itself is structurally easier due to the preprocessing. With a suitably selected expansion bound, we avoid flooding the solver with too large formulas.

Why are we not seeing the same behavior with Quantor? The reason is probably that Quantor also uses universal expansion, so there is no other technique that might be more suitable for expensive universals. In this case, the above hypothesis about expanding preferably the cheap universals does not apply. In fact, it appears that our refinements of universal expansion are indeed beneficial. Then it is certainly good if our preprocessor takes over as much as possible, and this is probably why the curve appears to go up even further for larger expansion bounds.

With a default expansion bound of $\beta_{exp} = 2.0$, we seem to be in an area which is generally beneficial to all solvers. We prefer to keep it as low as possible, since we think that a preprocessor should rather be defensive, and we do not want to overload less optimized solvers.

5.8. Summary

Making it easier for QBF^* solvers by selectively removing universal variables in a preprocessing step - a simple yet intriguing idea. We have successfully realized and experimentally verified this approach on the basis of universal expansion, which we have made bounded and more general by choosing universals from the whole prefix, or even miniscoped partitions of such universal scopes. In the scheduling of expansions, we include an element of goal orientation by also considering possible expansion benefits, such as the generation of unit literals.

We have observed that the size of an expanded formula directly depends on the set of existentially quantified variables which are dominated by the expanded universal, so we have placed a special focus on the problem of accurately determining these existentials. We have presented a dependency concept which is more compact than existing approaches by taking into account both local connectivity of variables and their polarity. These ideas are not only useful in the

context of universal expansion, and some of them have already been picked up in subsequent publications, e.g. [LB08a, LB08b]. By resolving specifically on dependent existentials, we further reduce the overhead of universal expansion.

Our experiments with three state-of-the-art *QBF* solvers and well-known benchmark problems from the QBFLIB have shown that the preprocessing often leads to considerable speedups and increases in the number of solved problems. On the other hand, the bounding appears to mostly prevent negative effects, so that the number of solved problems has never decreased in our experiments. Besides improvements in the solver performance, the experiments have also shown that our preprocessing is usually very effective at simplifying the prefixes, which greatly helps in making formulas structurally easier. It appears that universal expansion in combination with our refinements is the method of choice to handle universal quantifiers with small to moderate-size scopes.

6. Conclusion

Quantification is a powerful tool to obtain natural and compact encodings. We have, however, seen that not all usage patterns of quantifiers are equally effective, which makes it important to balance clarity and conciseness versus decision complexity. The contributions in this thesis support that tradeoff by providing

- relationships between formula structure and quantifier expressiveness,
- preprocessing that eliminates weak quantifiers,
- useful subclasses with lower complexity,
- and suitable modeling patterns.

One of our key observations is that there are close connections between the clause structure, the structure of models, and the universal expansion method. Our focus has been on universal expansion, because for *CNF* formulas, the elimination of universal instead of existential quantifiers appears to be generally cheaper (although still exponential in nature). We have shown that the imbalance between universal and existential quantification is particularly evident when the quantified variables satisfy the Horn property. In this case, all universal quantifiers can be expanded simultaneously with less than quadratic formula growth. What makes this result particularly interesting is that it even applies if unrestricted free variables are allowed. For such $QHORN^b$ formulas, it follows that $\exists HORN^b =_{poly-time} QHORN^b$. We have seen that these formulas are a natural representation for applications like *CNF* transformations or graph encodings. In addition to valuable insights into the behavior of quantifiers, our quadratic universal expansion can be applied by general QBF^* solvers to more efficiently handle $QHORN^b$ (sub)problems that may either be part of the input formula or arise after initial operations or branchings.

For arbitrary $QCNF^*$ formulas, we have shown that the arity of model functions can be restricted by computing transitive closures of local connectivity between

variables in common clauses in consideration of variable polarity. Our dependency concept is more compact than existing approaches, in the best case by an arbitrarily large factor. This has allowed us to significantly reduce the amount of copying required by universal expansion. It is clear that concise dependencies are also very important for other solving techniques like symbolic skolemization or QDPLL, e.g. in order to avoid unnecessary branchings.

Our dependency concept can recover non-prenex structures that have been hidden inevitably by prenexing transformations. As an alternative to recovering dependencies retroactively, we have considered dependency quantified Boolean formulas *DQBF**. The explicit indication of variable dependencies allows new modeling approaches which are more natural and more compact than ordinary *QBF** or even non-prenex *QBF** encodings. We have seen that the increased compactness is due to the ability to better reuse variables in order to save space, which has been illustrated by our new *DQBF** encoding of bounded reachability, an important subproblem of bounded model checking. Restrictions on the structure of the dependencies have led us to easier subclasses that avoid the *NEXPTIME*-completeness of arbitrary *DQBF**. In addition, the Horn property appears to be such a strong restriction on the clause structure that Horn formulas remain tractable even with dependency quantification. This makes *DQHORN** the first non-trivial class of formulas known to be tractable in combination with dependency quantification.

Based on the assumption that locally used quantifiers do not contribute much to the compactness of an encoding, yet can make solving the formulas much harder in practice, we have suggested a preprocessing for *QCNF** formulas. The idea is to eliminate as many of those cheap quantifiers as possible within given bounds, with a clear focus on the universal ones. Experiments with well-known problems from the QBFLIB formula collection have demonstrated that this preprocessing is very effective on simplifying the prefixes. On average, almost 50% of the universals and more than 25% of the existentials have been eliminated without increasing the formula length. Consequently, we have also seen significant performance improvements with state-of-the-art *QBF* solvers. In particular, the solver SQBF was able to solve almost 40% more problems in less than 30% of the time originally required.

The success of this preprocessing approach motivates further research on possible improvements. In particular, we assume that it is beneficial to not only eliminate variables, but also to introduce new ones when there are considerable

redundancies in the formula. The idea is to have two thresholds β_{elim} and β_{add} with $\beta_{elim} \ll \beta_{add}$, such that quantifiers are eliminated whenever the cost of doing so is below β_{elim}, and new quantifiers are introduced whenever this leads to a compression of more than β_{add}. In the latter case, some of the encoding rules from Section 2.6 could be used. The easiest is probably the abbreviation of repeating subformulas with new existential variables. A tool for automatically applying such encoding rules to arbitrary propositional or quantified Boolean formulas would also be interesting on its own. A main problem, however, is the efficient detection of repetition or similarity. Another interesting improvement would be to resolve not only on existentials from the innermost quantifier block. This could be accomplished without compromising soundness by taking into account the dependency information that is already available. In addition, we would like to evaluate the inclusion of further strategic elements into our variable selection. For example, it might be interesting to give preference to universals that appear in short clauses, or to attempt making local areas of the formula completely free of universals.

Determining compact variable dependencies is not only an essential part of our preprocessing, but it is also an important problem for QBF^* solving in general. We see the possibility to make our dependency concept even more concise by considering in addition to variable polarity also complementary universals, similar to the idea of universal blocking in Q-resolution. The dependency computation itself might also be optimized, e.g. by taking advantage of prior miniscoping and by using more sophisticated data structures for computing and storing dependencies, like those suggested in [LB08b].

For $DQBF^*$, we have demonstrated that universal expansion can easily be lifted from QBF^*. Additional solving techniques would be required to build a reasonably powerful $DQBF^*$ solver. Symbolic skolemization as it is used in sKizzo [Ben05a] can probably be adapted in a natural way to dependency quantification, but it is unclear how Q-resolution could be lifted to $DQBF^*$. In the QBF^* case, the refutation completeness of Q-resolution is shown by successively eliminating existentials from innermost to outermost, a technique which is not applicable to partially ordered quantifiers.

Our expressiveness result $\exists HORN^b =_{poly-time} QHORN^b =_{poly-time} DQHORN^b$ complements the existing result $CNF <_{poly-length} \exists HORN^b$. We assume that also $PROP <_{poly-length} \exists HORN^b$. Since $PROP \leq_{poly-time} \exists 2\text{-}HORN^b$ by our PS-graph CNF transformation from Section 3.8, it would be sufficient to prove

$\exists$2-*HORN*^{b} $<_{poly-length}$ $\exists$*HORN*^{b}, or *PROP* $<_{poly-length}$ $\exists$2-*HORN*^{b}. Still, this appears to be a very challenging problem, in particular if we consider that such a result would imply Boolean circuits with arbitrary fan-out to be exponentially more powerful than circuits with fan-out 1, a problem which has been open for many years.

A. Overview of Formula Classes

For quick reference, this appendix provides a collection of the definitions of all formula classes which are considered in this document.

A.1. Propositional Formulas and Normal Forms $PROP, NNF, (k\text{-})CNF, (k\text{-})DNF$

With *PROP*, we denote the class of *propositional formulas*, which is inductively defined by the following rules (exclusively):

1. Every propositional variable is a formula.
2. If ϕ is a formula, its negation $(\neg\phi)$ is also a formula.
3. If ϕ_1 and ϕ_2 are formulas, the conjunction $(\phi_1 \wedge \phi_2)$ and the disjunction $(\phi_1 \vee \phi_2)$ are also formulas.

We furthermore allow implication $(\phi_1 \to \phi_2)$ as an abbreviation for $((\neg\phi_1) \vee \phi_2)$ and equivalence $(\phi_1 \leftrightarrow \phi_2)$ to denote $((\phi_1 \to \phi_2) \wedge (\phi_1 \leftarrow \phi_2))$.
If parentheses are omitted, we assume the following operator binding priorities: $\neg$ has highest priority, followed in decreasing order by $\wedge$, $\vee$, $\to$ and finally $\leftrightarrow$.

NNF is the class of propositional formulas in *negation normal form*, which requires that negations only occur immediately in front of a variable. That means a *NNF* formula is either a positive propositional variable x, a negated variable $(\neg x)$, or a conjunction $(\phi_1 \wedge \phi_2)$ or disjunction $(\phi_1 \vee \phi_2)$ of *NNF* formulas ϕ_1, ϕ_2.

A propositional formula is in *conjunctive normal form* (*CNF*) if it is of the form

$$\phi = \bigwedge_{i=1}^{q} \bigvee_{j=1}^{s_i} l_{i,j}$$

where a literal $l_{i,j}$ is a positive or negated propositional variable.

Analogously, formulas in *disjunctive normal form* (*DNF*) have the form

$$\psi = \bigvee_{i=1}^{q} \bigwedge_{j=1}^{s_i} l_{i,j}$$

for literals $l_{i,j}$.

For $k \geq 1$, *k-CNF* (*k-DNF*, respectively) contains all *CNF* (*DNF*) formulas with at most k literals per clause, i.e. $s_i \leq k$ in the formulas above.

A.2. Quantified Boolean Formulas $QBF^{(*)}, \exists BF^{(*)}, Q(k\text{-})CNF^{(*)}, Q(k\text{-})DNF^{(*)}$

A *quantified Boolean formula* $\Phi \in QBF$ (in prenex form) is a formula of the form

$$\Phi = Q_1 v_1 ... Q_k v_k \ \phi(v_1, ..., v_k)$$

with a prefix of existential or universal quantifiers $Q_i \in \{\exists, \forall\}$ and propositional variables v_i. The matrix ϕ is a propositional formula over the quantified variables.

With QBF^*, we denote *quantified Boolean formulas with free variables*, that means formulas

$$\Psi(z_1, ..., z_r) = Q_1 v_1 ... Q_k v_k \ \psi(v_1, ..., v_k, z_1, ..., z_r)$$

where the propositional matrix may also contain variables which are not bound by a quantifier.

QBF or QBF^* formulas with a purely existential prefix ($Q_1 = \exists, ..., Q_k = \exists$) are called *existentially quantified Boolean formulas* $\exists BF$ (without free variables) or $\exists BF^*$ (with free variables).

Let $K \in \{NNF, CNF, k\text{-}CNF, DNF, k\text{-}DNF\}$ be a class of propositional formulas. Then QK (QK^* if free variables are allowed) denotes the corresponding subclass of *QBF* (QBF^*, respectively) that contains all formulas whose matrix is in K.

A.3. Horn Formulas

The Horn property is the syntactic restriction of having at most one positive literal per clause. A variety of interesting formula classes can be formed around this property. They differ in the kinds of variables which we allow and in whether the Horn property is enforced on all allowed variables or only a subset of these. Table A.1 provides an overview of important formula classes, and the following sections provide the corresponding formal definitions.

Table A.1.: Overview of Horn Formula Classes

Formula class	Allowed variable types	Horn property applies to
HORN	free	free
QHORN	$\exists$, $\forall$	$\exists$, $\forall$
QEHORN	$\exists$, $\forall$	$\exists$
$QHORN^*$	$\exists$, $\forall$, free	$\exists$, $\forall$, free
$QHORN^b$	$\exists$, $\forall$, free	$\exists$, $\forall$
$QEHORN^*$	$\exists$, $\forall$, free	$\exists$, free
$\exists HORN$	$\exists$	$\exists$
$\exists HORN^*$	$\exists$, free	$\exists$, free
$\exists HORN^b$	$\exists$, free	$\exists$

A.3.1. Propositional Classes *HORN*, *k-HORN*

The class of *propositional Horn formulas* (*HORN*) contains all *CNF* formulas with at most one positive literal per clause. For a constant $k \geq 2$, *k-HORN* means *HORN* formulas with at most k literals per clause.

A.3.2. Quantified Horn Formulas $Q(k\text{-})HORN, Q(k\text{-})HORN^*$

With *QHORN* (*Qk-HORN*, respectively), we denote *quantified Horn formulas*, that means formulas

$$\Phi = Q_1 v_1 ... Q_k v_k \ \phi(v_1, ..., v_k)$$

where $\phi \in HORN$ (k-*HORN*, respectively).

Formulas

$$\Psi(z_1, ..., z_r) = Q_1 v_1 ... Q_k v_k \ \psi(v_1, ..., v_k, z_1, ..., z_r)$$

with free variables which also satisfy the Horn property, that is $\psi \in (k\text{-})HORN$, are called *quantified (k-)Horn formulas with free variables*, denoted by $QHORN^*$ ($Qk\text{-}HORN^*$, respectively).

A.3.3. Generalized Horn $Q(k\text{-})HORN^b, Q(k\text{-})EHORN^{(*)}$

Generalizations of quantified Horn formulas arise when the Horn property is enforced only on certain kinds of variables. One possibility is to allow free variables that are exempt from the Horn property. Then $Q(k\text{-})HORN^b$ is the class of all formulas

$$\Gamma(z_1, ..., z_r) = Q_1 v_1 ... Q_k v_k \ \gamma(v_1, ..., v_k, z_1, ..., z_r) \in QCNF^*$$

with $\gamma \in (k\text{-})HORN$ after removing all literals over free variables.

Another possibility is to enforce the Horn property only on the existential and free variables. Accordingly, $Q(k\text{-})EHORN^*$ contains all formulas

$$\Omega(z_1, ..., z_r) = Q_1 v_1 ... Q_k v_k \ \omega(v_1, ..., v_k, z_1, ..., z_r) \in QCNF^*$$

for which $\omega \in (k\text{-})HORN$ after removing all universal literals. As before, we omit the star * and write $Q(k\text{-})EHORN$ if only formulas without free variables are considered.

A.3.4. Existential Prefix $\exists(k\text{-})HORN^{(*)}, \exists(k\text{-})HORN^b$

With $\exists(k\text{-})HORN$ ($\exists(k\text{-})HORN^*$, $\exists(k\text{-})HORN^b$, respectively), we denote the subclass of $Q(k\text{-})HORN$ ($Q(k\text{-})HORN^*$, $Q(k\text{-})HORN^b$, respectively) with purely existential prefix.

A.3.5. Renamings $ren\text{-}Q(k\text{-})(E)HORN^{(*)}, ren\text{-}Q(k\text{-})HORN^{b}$

Let $QK \in \{Q(k\text{-})HORN^{(*)}, Q(k\text{-})HORN^{b}, Q(k\text{-})EHORN^{(*)}\}$ be one of the previously defined classes of (generalized) quantified Horn formulas. With *ren-QK*, we denote formulas Π for which there exists a renaming of literals, i.e. a function $f : literals(\Pi) \to literals(\Pi)$ with $f(l) \in \{\neg l, l\}$ and $f(\neg l) \approx \neg f(l)$ for all $l \in literals(\Pi)$, such that $\Pi[l/f(l)] \in QK$ when the renaming is applied to all literals.

A.4. Dependency Quantified Formulas

A.4.1. Base Classes $DQBF, DQBF^{*}$

A *dependency quantified Boolean formula* $\Phi \in DQBF$ with universal variables $\mathbf{x} = (x_1, ..., x_n)$ and existential variables $\mathbf{y} = (y_1, ..., y_m)$ $(n, m \geq 0)$ is a formula of the form

$$\Phi = \forall x_1 ... \forall x_n \exists y_1 (x_{d_{1,1}}, ..., x_{d_{1,n_1}}) ... \exists y_m (x_{d_{m,1}}, ..., x_{d_{m,n_m}})\ \phi(\mathbf{x}, \mathbf{y})$$

where dependency lists $\mathbf{x}_{\mathbf{d}_i} := (x_{d_{i,1}}, ..., x_{d_{i,n_i}})$, $1 \leq d_{i,j} \leq n$, indicate the universal variables $x_{d_{i,j}}$ on which an existential variable y_i depends. The matrix ϕ is a propositional formula over the quantified variables.

As before, a star * indicates that free variables are allowed, so $DQBF^{*}$ is the class of formulas

$$\Psi = \forall x_1 ... \forall x_n \exists y_1 (x_{d_{1,1}}, ..., x_{d_{1,n_1}}) ... \exists y_m (x_{d_{m,1}}, ..., x_{d_{m,n_m}})\ \psi(\mathbf{x}, \mathbf{y}, \mathbf{z})$$

with a matrix ψ over universal variables $\mathbf{x} = (x_1, ..., x_n)$, existential variables $\mathbf{y} = (y_1, ..., y_m)$ and free variables $\mathbf{z} = (z_1, ..., z_r)$.

A.4.2. Normal Forms $DQNNF^{(*)}, DQ(k\text{-})CNF^{(*)}, DQ(k\text{-})DNF^{(*)}$

Let $K \in \{NNF, CNF, k\text{-}CNF, DNF, k\text{-}DNF\}$ be a propositional normal form. By DQK (DQK^{*} if free variables are allowed), we denote the corresponding subclass of $DQBF$ ($DQBF^{*}$, respectively) that contains all formulas whose matrix is in K.

A.4.3. Dependency Quantified Horn Formulas $DQ(k\text{-})HORN^{(*)}, DQ(k\text{-})HORN^{b}$

A *dependency quantified (k-)Horn formula* $\Phi \in DQ(k\text{-})HORN$ with universal variables $\mathbf{x} = (x_1, ..., x_n)$ and existential variables $\mathbf{y} = (y_1, ..., y_m)$ $(n, m \geq 0)$ is a formula of the form

$$\Phi = \forall x_1 ... \forall x_n \exists y_1(\mathbf{x}_{\mathbf{d}_1}) ... \exists y_m(\mathbf{x}_{\mathbf{d}_m})\ \phi(\mathbf{x}, \mathbf{y})$$

with $\phi \in (k\text{-})HORN$. We write $DQ(k\text{-})HORN^{*}$ if ϕ can contain free variables.

$DQ(k\text{-})HORN^{b}$ denotes formulas of the form

$$\Psi = \forall x_1 ... \forall x_n \exists y_1(\mathbf{x}_{\mathbf{d}_1}) ... \exists y_m(\mathbf{x}_{\mathbf{d}_m})\ \psi(\mathbf{x}, \mathbf{y}, \mathbf{z})$$

with universal variables $\mathbf{x} = (x_1, ..., x_n)$, existential variables $\mathbf{y} = (y_1, ..., y_m)$ and free variables $\mathbf{z} = (z_1, ..., z_r)$, such that $\psi \in (k\text{-})HORN$ after removing all literals over free variables.

A.4.4. Restrictions on the Structure of Dependencies $D_{po}QBF^{(*)}, D_{log}QBF^{(*)}, D_{k}QBF^{(*)}$

$D_{po}QBF^{*}$ is the class of dependency quantified Boolean formulas

$$\Phi(\mathbf{z}) = \forall x_1 ... \forall x_n \exists y_1(\mathbf{x}_{\mathbf{d}_1}) ... \exists y_m(\mathbf{x}_{\mathbf{d}_m})\ \phi(\mathbf{x}, \mathbf{y}, \mathbf{z})$$

with *polynomially orderable dependencies*, that means

$$\left| \bigcup_{1 \leq j < k \leq m} (\mathbf{x}_{\mathbf{d}_{s_j}} \setminus \mathbf{x}_{\mathbf{d}_{s_k}}) \right| \in O(\log |\Phi|)$$

for some ordering $S = (s_1, ..., s_m) \in \{1, ..., m\}^m$ on the dependencies.

With $D_{log}QBF^{*}$, we denote the class of dependency quantified Boolean formulas with *logarithmic dependencies*. That is, formulas

$$\Psi(\mathbf{z}) = \forall x_1 ... \forall x_n \exists y_1(x_{d_{1,1}}, ..., x_{d_{1,n_1}}) ... \exists y_m(x_{d_{m,1}}, ..., x_{d_{m,n_m}})\ \psi(\mathbf{x}, \mathbf{y}, \mathbf{z})$$

with $n_1, ..., n_m \in O(log\ |\Psi|)$, such that each existential variable depends on at most logarithmically many universals.

For fixed $k \geq 0$, $D_kQBF^* \subseteq D_{log}QBF^*$ is the class of dependency quantified Boolean formulas in which each existential variable can depend on at most k universal variables. That means $n_1, ..., n_m \leq k$ for formulas with a prefix of $\forall x_1 ... \forall x_n \exists y_1 (x_{d_{1,1}}, ..., x_{d_{1,n_1}}) ... \exists y_m (x_{d_{m,1}}, ..., x_{d_{m,n_m}})$.

Again, we omit the star * and write $D_{po}QBF$, $D_{log}QBF$ or D_kQBF, respectively, if free variables are excluded.

B. Abstract

Quantified Boolean formulas (*QBF* or QBF^* with free variables) generalize propositional logic by allowing variables to be quantified universally or existentially. This enhancement allows problems from many areas, such as planning or verification, to be encoded in a natural way. The resulting formulas are often significantly more compact than their propositional equivalents, but also more difficult to solve. Not all usage patterns of quantifiers seem equally effective, which makes it important to balance clarity and conciseness versus decision complexity. The contributions in this thesis support that tradeoff by providing

- relationships between formula structure and quantifier expressiveness,
- preprocessing that eliminates weak quantifiers,
- useful subclasses with lower complexity,
- and suitable modeling patterns.

We show that there exist close connections between the clause structure, the structure of Boolean function models, which describe the behavior of quantifiers, and the method of universal quantifier expansion for simplifying prefixes.

In particular, we consider quantified Horn formulas (*QHORN*) and generalizations thereof. For these formulas, we prove that the behavior of the existential quantifiers depends only on the cases where at most one of the universally quantified variables is zero, which allows a transformation into short purely existentially quantified formulas and the computation of satisfiability models in time $O(|\forall| \cdot |\Phi|)$ for formulas of length $|\Phi|$ with $|\forall|$ universal quantifiers. We extend our results to formulas with free variables ($QHORN^*$) and show that they have monotone equivalence models. A further extension is possible to clausal formulas with free variables where only the quantified variables satisfy the Horn property. We prove that any such formula $\Phi \in QHORN^b$ can be transformed in time $O(|\forall| \cdot |\Phi|)$ into an equivalent formula of length $O(|\forall| \cdot |\Phi|)$ which has only existential quantifiers. It is demonstrated that these formulas can be used

to encode graph structures and propositional *CNF* transformations in a natural way, which leads to a new compact graph-based *CNF* transformation. It nicely preserves and visualizes the structure of the propositional input formula and requires only two existentially quantified variables per clause ($\exists 2\text{-}HORN^b$).

For arbitrary *QCNF* or $QCNF^*$ formulas, we show that the arity of model functions can be restricted by computing transitive closures of local connectivity between variables in common clauses in consideration of variable polarity. Our dependency concept is more compact than existing approaches, in the best case by an arbitrarily large factor. This allows us to significantly reduce the amount of copying required by universal expansion and makes it feasible to expand quantifiers from arbitrary scopes. On the basis of these expansion refinements, we present a preprocessing of $QCNF^*$ formulas by expanding a selection of universal variables with bounded expansion costs. We describe a suitable selection strategy which combines cost estimates with goal orientation, and we also integrate Q-resolution specifically to further reduce the costs of expansion steps. Experiments with well-known problems from the QBFLIB formula collection demonstrate that this preprocessing is very effective on simplifying the prefixes and can significantly improve the performance of state-of-the-art *QBF* solvers.

An alternative to recovering variable dependencies from given formulas is to extend quantified Boolean formulas with explicitly given dependencies. Such dependency quantified Boolean formulas (*DQBF* or $DQBF^*$) allow novel and even more concise modeling patterns, which we demonstrate with a new encoding of the well-known bounded reachability problem for directed graphs. For 2^n vertices, it requires only $O(n)$ variables, in contrast to $O(n^2)$ (but in an unbounded number of quantifier blocks) or even $O(2^n)$ in existing QBF^* approaches.

While dependency quantification generally causes an increase in complexity, we consider easier subclasses with restrictions on the prefix structure. We show that formulas with dependencies of bounded or at most logarithmic size (D_kQBF^* or $D_{log}QBF^*$) have Σ_2^P-complete satisfiability problems, and formulas with polynomially orderable dependencies ($D_{po}QBF^*$) are presented as a *PSPACE*-complete generalization of QBF^*. We also show that important techniques like universal quantifier expansion can be lifted naturally to these formulas and that Horn formulas remain tractable with dependency quantification. It follows that dependency quantified Horn formulas with unrestricted free variables ($DQHORN^b$) are as expressive as $QHORN^b$ or purely existentially quantified $\exists HORN^b$ formulas: $DQHORN^b =_{poly-time} QHORN^b =_{poly-time} \exists HORN^b$.

References

[AB81] S. Anderaa and E. Börger. *The Equivalence of Horn and Network Complexity for Boolean Functions*. Acta Informatica, 15:303–307, 1981.

[AB02] A. Ayari and D. Basin. *QUBOS: Deciding Quantified Boolean Logic using Propositional Satisfiability Solvers*. Proc. 4th Intl. Conf. on Formal Methods in Computer-Aided Design (FMCAD 2002), Springer LNCS 2517, pp. 187–201, 2002.

[Asp80] B. Aspvall. *Recognizing Disguised NR(1) Instances of the Satisfiability Problem*. Journal of Algorithms, 1(1):97–103, 1980.

[ASV+05] M. Ali, S. Safarpour, A. Veneris, M. Abadir, and R. Drechsler. *Post-Verification Debugging of Hierarchical Designs*. Proc. 6th Intl. Workshop on Microprocessor Test and Verification (MTV 2005), pp. 42–47, 2005.

[BBF+73] M. Bauer, D. Brand, M. Fischer, A. Meyer, and M. Paterson. *A Note on Disjunctive Form Tautologies*. SIGACT News, 5(2):17–20, 1973.

[BCCZ99] A. Biere, A. Cimatti, E. Clarke, and Y. Zhu. *Symbolic Model Checking without BDDs*. Proc. 5th Intl. Conf. on Tools and Algorithms for Construction and Analysis of Systems (TACAS 1999), Springer LNCS 1579, pp. 193–207, 1999.

[Ben05a] M. Benedetti. *Evaluating QBFs via Symbolic Skolemization*. Proc. 11th Intl. Conf. on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR 2004), Springer LNCS 3452, pp. 285–300, 2005.

[Ben05b] M. Benedetti. *Quantifier Trees for QBFs*. Proc. 8th Intl. Conf. on Theory and Applications of Satisfiability Testing (SAT 2005), Springer LNCS 3569, pp. 378–385, 2005.

[BFL91] L. Babai, L. Fortnow, and C. Lund. *Non-Deterministic Exponential Time has Two-Prover Interactive Protocols*. Journal Computational Complexity, 1(1):3–40, 1991.

[BG86] A. Blass and Y. Gurevich. *Henkin Quantifiers and Complete Problems*. Annals of Pure and Applied Logic, 32(1):1–16, 1986.

[Bie05] A. Biere. *Resolve and Expand*. Proc. 7th Intl. Conf. on Theory and Applications of Satisfiability Testing (SAT 2004), Revised Selected Papers, Springer LNCS 3542, pp. 59–70, 2005.

[BKB06] U. Bubeck and H. Kleine Büning. *Dependency Quantified Horn Formulas: Models and Complexity*. Proc. 9th Intl. Conf. on Theory and Applications of Satisfiability Testing (SAT 2006), Springer LNCS 4121, pp. 198–211, 2006.

[BKB07] U. Bubeck and H. Kleine Büning. *Bounded Universal Expansion for Preprocessing QBF*. Proc. 10th Intl. Conf. on Theory and Applications of Satisfiability Testing (SAT 2007), Springer LNCS 4501, pp. 244–257, 2007.

[BKB08] U. Bubeck and H. Kleine Büning. *Models and Quantifier Elimination for Quantified Horn Formulas*. Journal Discrete Applied Mathematics, 156(10):1606–1622, 2008.

[BKB09] U. Bubeck and H. Kleine Büning. *A New 3-CNF Transformation by Parallel-Serial Graphs*. Journal Information Processing Letters, 109(7):376–379, 2009.

[BKBZ05] U. Bubeck, H. Kleine Büning, and X. Zhao. *Quantifier Rewriting and Equivalence Models for Quantified Horn Formulas*. Proc. 8th Intl. Conf. on Theory and Applications of Satisfiability Testing (SAT 2005), Springer LNCS 3569, pp. 386–392, 2005.

[BOGKW88] M. Ben-Or, S. Goldwasser, J. Kilian, and A. Widgerson. *Multi-Prover Interactive Proofs: How to Remove Intractability Assumptions*. Proc. 20th Ann. ACM Symp. Theory of Computing, pp. 113–131, 1988.

[Bub03] U. Bubeck. *Design of a Modular Platform for Automated Theorem Proving in Multiple Logics*. M.S. Thesis, San Diego State University, San Diego, USA, 2003.
Software available at http://www.ub-net.de/cms/proverbox.html.

[CA93] J. Crawford and L. Auton. *Experimental Results on the Crossover Point in Satisfiability Problems*. Proc. 11th Natl. Conf. on Artificial Intelligence (AAAI 1993), pp. 21–27, 1993.

[CBRZ01] E. Clarke, A. Biere, R. Raimi, and Y. Zhu. *Bounded Model Checking Using Satisfiability Solving*. Formal Methods in System Design, 19(1):7–34, 2001.

[CD05] H. Chen and V. Dalmau. *Looking Algebraically at Tractable Quantified Boolean Formulas*. Proc. 7th Intl. Conf. on Theory and Applications of Satisfiability Testing (SAT 2004), Revised Selected Papers, Springer LNCS 3542, pp. 71–79, 2005.

[CGS97] M. Cadoli, A. Giovanardi, and M. Schaerf. *Experimental Analysis of the Computational Cost of Evaluating Quantified Boolean Formulae*. Proc. 5th Congr. of the Italian Association for Artificial Intelligence (AI*IA 1997), Springer LNAI 1321, pp. 207–218, 1997.

[CI05] H. Chen and Y. Interian. *A Model for Generating Random Quantified Boolean Formulas*. Proc. 19th Intl. Joint Conf. on Artificial Intelligence (IJCAI 2005), pp. 66–71, 2005.

[CKS01] N. Creignou, S. Khanna, and M. Sudan. *Complexity Classifications of Boolean Constraint Satisfaction Problems*. SIAM Monographs on Discrete Applied Mathematics, 2001.

[CMLBL05] S. Coste-Marquis, D. Le Berre, and F. Letombe. *A Branching Heuristics for Quantified Renamable Horn Formulas*. Proc. 8th Intl. Conf. on Theory and Applications of Satisfiability Testing (SAT 2005), Springer LNCS 3569, pp. 393–399, 2005.

[CSGG02] M. Cadoli, M. Schaerf, A. Giovanardi, and M. Giovanardi. *An Algorithm to Evaluate Quantified Boolean Formulae and Its Experimental Evaluation*. Journal of Automated Reasoning, 28(2):101–142, 2002.

[Dal97] V. Dalmau. *Some Dichotomy Theorems on Constant-free Boolean Formulas*. Technical Report TR-LSI-97-43-R, Universitat Polytechnica de Catalunya, 1997.

[DBC01] P. Dunne and T. Bench-Capon. *A Sharp Threshold for the Phase Transition of a Restricted Satisfiability Problem for Horn Clauses*. Journal of Logic and Algebraic Programming, 47(1):1–14, 2001.

[DG84] W. Dowling and J. Gallier. *Linear-Time Algorithms for Testing the Satisfiability of Propositional Horn Formulae*. Journal of Logic Programming, 1(3):267–284, 1984.

[DHK05] N. Dershowitz, Z. Hanna, and J. Katz. *Bounded Model Checking with QBF*. Proc. 8th Intl. Conf. on Theory and Applications of Satisfiability Testing (SAT 2005), Springer LNCS 3569, pp. 408–414, 2005.

[DLL62] M. Davis, G. Logemann, and D. Loveland. *A Machine Program for Theorem Proving*. Communications of the ACM, 5(7):394–397, 1962.

[DP60] M. Davis and H. Putnam. *A Computing Procedure for Quantification Theory*. Journal of the ACM, 7(3):201–215, 1960.

[EST+04] U. Egly, M. Seidl, H. Tompits, S. Woltran, and M. Zolda. *Comparing Different Prenexing Strategies for Quantified Boolean Formulas*. Proc. 6th Intl. Conf. on Theory and Applications of Satisfiability Testing (SAT 2003), Revised Selected Papers, Springer LNCS 2919, pp. 214–228, 2004.

[ETW02] U. Egly, H. Tompits, and S. Woltran. *On Quantifier Shifting for Quantified Boolean Formulas*. Proc. SAT 2002 Workshop on Theory and Applications of Quantified Boolean Formulas (QBF 2002), pp. 48–61, 2002.

[FKKB90] A. Flögel, M. Karpinski, and H. Kleine Büning. *Subclasses of Quantified Boolean Formulas*. Proc. 4th Workshop on Computer Science Logic (CSL 1990), Springer LNCS 533, pp. 145–155, 1990.

[FKKB95] A. Flögel, M. Karpinski, and H. Kleine Büning. *Resolution for Quantified Boolean Formulas*. Information and Computation, 117(1):12–18, 1995.

[FKS+04] J. Franco, M. Kouril, J. Schlipf, J. Ward, S. Weaver, M. Dransfield, and M. Vanfleet. *SBSAT: a State-Based, BDD-Based Satisfiability Solver*. Proc. 6th Intl. Conf. on Theory and Applications of Satisfiability Testing (SAT 2003), Revised Selected Papers, Springer LNCS 2919, pp. 398–410, 2004.

[FMS00] R. Feldmann, B. Monien, and S. Schamberger. *A Distributed Algorithm to Evaluate Quantified Boolean Formulae*. Proc. 17th Natl. Conf. on Artificial Intelligence (AAAI 2002), pp. 285–290, 2000.

[GGN+04] I. Gent, E. Giunchiglia, M. Narizzano, A. Rowley, and A. Tacchella. *Watched Data Structures for QBF Solvers*. Proc. 6th Intl. Conf. on Theory and Applications of Satisfiability Testing (SAT 2003), Revised Selected Papers, Springer LNCS 2919, pp. 25–36, 2004.

[GNT01] E. Giunchiglia, M. Narizzano, and A. Tacchella. *Quantified Boolean Formulas Satisfiability Library (QBFLIB)*. Website http://www.qbflib.org, 2001.

[GNT02] E. Giunchiglia, M. Narizzano, and A. Tacchella. *Learning for Quantified Boolean Logic Satisfiability*. Proc. 18th Natl. Conf. on Artificial Intelligence (AAAI 2002), pp. 649–654, 2002.

[GNT07] E. Giunchiglia, M. Narizzano, and A. Tacchella. *Quantifier Structure in Search-Based Procedures for QBFs*. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 26(3):497–507, 2007.

[Grä92] E. Grädel. *Capturing Complexity Classes by Fragments of Second Order Logic*. Theoretical Computer Science, 101(1):35–57, 1992.

[GW84] H. Galperin and A. Widgerson. *Succinct representations of graphs*. Information and Control, 56(3):183–198, 1984.

[GW99] I. Gent and T. Walsh. *Beyond NP: The QSAT Phase Transition*. Proc. 16th Natl. Conf. on Artificial Intelligence (AAAI 1999), pp. 648–653, 1999.

[Hen61] L. Henkin. *Some remarks on infinitely long formulas*. In: Infinitistic Methods (Warsaw, 1961), pp. 167–183, 1961.

[Hor51] A. Horn. *On sentences which are true of direct unions of algebras*. Journal of Symbolic Logic, 16(1):14–21, 1951.

[Héb94] J. Hébrard. *A Linear Algorithm for Renaming a Set of Clauses as a Horn Set*. Theoretical Computer Science, 124(2):343–350, 1994.

[Imm99] N. Immerman. *Descriptive Complexity*. Graduate Texts in Computer Science, Springer, New York, 1999.

[Ist02] G. Istrate. *The Phase Transition in Random Horn Satisfiability and its Algorithmic Implications*. Random Structures and Algorithms, 20(4):483–506, 2002.

[JB07] T. Jussila and A. Biere. *Compressing BMC Encodings with QBF*. Electronic Notes in Theoretical Computer Science, 174(3):45–56, 2007.

[JS05] H. Jin and F. Somenzi. *CirCUs: A Hybrid Satisfiability Solver*. Proc. 7th Intl. Conf. on Theory and Applications of Satisfiability Testing (SAT 2004), Revised Selected Papers, Springer LNCS 3542, pp. 211–223, 2005.

[KBB09] H. Kleine Büning and U. Bubeck. *Theory of Quantified Boolean Formulas*. In: Handbook of Satisfiability, A. Biere, M. Heule, H. van Maaren, and T. Walsh (Eds.), IOS Press, pp. 735–760, 2009.

[KBL99] H. Kleine Büning and T. Lettmann. *Propositional Logic: Deduction and Algorithms*. Cambridge University Press, Cambridge, UK, 1999.

[KBSZ04] H. Kleine Büning, K. Subramani, and X. Zhao. *On Boolean Models for Quantified Boolean Horn Formulas*. Proc. 6th Intl. Conf. on Theory and Applications of Satisfiability Testing (SAT 2003), Revised Selected Papers, Springer LNCS 2919, pp. 93–104, 2004.

[KBSZ07] H. Kleine Büning, K. Subramani, and X. Zhao. *Boolean Functions as Models for QBF*. Journal of Automated Reasoning, 39(1):49–75, 2007.

[KBZ05] H. Kleine Büning and X. Zhao. *Equivalence Models for Quantified Boolean Formulas*. Proc. 7th Intl. Conf. on Theory and Applications of Satisfiability Testing (SAT 2004), Revised Selected Papers, Springer LNCS 3542, pp. 224–234, 2005.

[KBZB09] H. Kleine Büning, X. Zhao, and U. Bubeck. *Resolution and Expressiveness of Subclasses of Quantified Boolean Formulas and Circuits*. Proc. 12th Intl. Conf. on Theory and Applications of Satisfiability Testing (SAT 2009), Springer LNCS 5584, pp. 391–397, 2009.

[KKBS87] M. Karpinski, H. Kleine Büning, and P. Schmitt. *On the computational complexity of quantified Horn clauses*. Proc. 1st Workshop on Computer Science Logic (CSL 1987), Springer LNCS 329, pp. 129–137, 1987.

[KS92] H. Kautz and B. Selman. *Planning as Satisfiability*. Proc. 10th European Conf. on Artificial Intelligence (ECAI 1992), pp. 359–363, 1992.

[KS96] H. Kautz and B. Selman. *Pushing the Envelope: Planning, Propositional Logic, and Stochastic Search*. Proc. 13th Natl. Conf. on Artificial Intelligence (AAAI 1996), pp. 1194–1201, 1996.

[LB08a] F. Lonsing and A. Biere. *Nenofex: Expanding NNF for QBF Solving*. Proc. 11th Intl. Conf. on Theory and Applications of Satisfiability Testing (SAT 2008), Springer LNCS 4996, pp. 196–210, 2008.

[LB08b] F. Lonsing and A. Biere. *Efficiently Representing Existential Dependency Sets for Expansion-based QBF Solvers*. Proc. 4th Doctoral Workshop on Mathematical and Engineering Methods in Computer Science (MEMICS 2008), 2008.

[Mac92] P. Macmahon. *The Combination of Resistances*. The Electrician, 28:601–602, 1892.

[MMZ+01] M. Moskewicz, C. Madigan, Y. Zhao, L. Zhang, and S. Malik. *Chaff: Engineering an Efficient SAT Solver*. Proc. 38th Design Automation Conference (DAC 2001), pp. 530–535, 2001.

[MS72] A. Meyer and L. Stockmeyer. *The Equivalence Problem for Regular Expressions with Squaring Requires Exponential Space*. Proc. 13th Symp. on Switching and Automata Theory, pp. 125–129, 1972.

[MS73] A. Meyer and L. Stockmeyer. *Word Problems Requiring Exponential Time*. Preliminary Report, Proc. 5th ACM Symp. on Theory of Computing (STOC 1973), pp. 1–9, 1973.

[MS04] M. Mneimneh and K. Sakallah. *Computing Vertex Eccentricity in Exponentially Large Graphs: QBF Formulation and Solution*. Proc. 6th Intl. Conf. on Theory and Applications of Satisfiability Testing (SAT 2003), Revised Selected Papers, Springer LNCS 2919, pp. 411–425, 2004.

[MSS96] J. Marques-Silva and A. Sakallah. *Grasp: A New Search Algorithm for Satisfiability*. Proc. 1996 IEEE/ACM Intl. Conf. on Computer-Aided Design (ICCAD 1996), pp. 220–227, 1996.

[NW01] A. Nonnengart and C. Weidenbach. *Computing Small Clause Normal Forms*. In: Handbook of Automated Reasoning, A. Robinson, A. Voronkov (Eds.), Elsevier, pp. 335–367, 2001.

[Pap94] C. Papadimitriou. *Computational Complexity*. Addison-Wesley Publishing, Reading, Mass., 1994.

[PG86] D. Plaisted and S. Greenbaum. *A Structure-preserving Clause Form Translation*. Journal of Symbolic Computation, 2(3):293–304, 1986.

[PR79] G. Peterson and J. Reif. *Multiple-Person Alternation*. Proc. 20th IEEE Symp. on Foundations of Computer Science, pp. 348–363, 1979.

[PRA01] G. Peterson, J. Reif, and S. Azhar. *Lower Bounds for Multiplayer Non-Cooperative Games of Incomplete Information*. Computers and Mathematics with Applications, 41(7-8):957–992, 2001.

[Rin99] J. Rintanen. *Improvements to the Evaluation of Quantified Boolean Formulae*. Proc. 16th Intl. Joint Conf. on Artificial Intelligence (IJCAI 1999), pp. 1192–1197, 1999.

[Sav70] W. Savitch. *Relationships Between Nondeterministic and Deterministic Tape Complexities*. Journal of Computer and System Sciences, 4(2):177–192, 1970.

[SB05] H. Samulowitz and F. Bacchus. *Using SAT in QBF*. Proc. 11th Intl. Conf. on Principles and Practice of Constraint Programming (CP 2005), Springer LNCS 3709, pp. 578–592, 2005.

[SC85] A. Sistla and E. Clarke. *The complexity of propositional linear time logics*. Journal of the ACM, 32(3):733–749, 1985.

[Sch78] T. Schaefer. *The complexity of satisfiability problems*. Proc. 10th ACM Symp. on Theory of Computing (STOC 1978), pp. 1–9, 1978.

[Sha38] C. Shannon. *A Symbolic Analysis of Relay and Switching Circuits*. Transactions of the American Institute of Electrical Engineers, 57:713–723, 1938.

[Sip05] M. Sipser. *Introduction to the Theory of Computation, 2nd edition*. Thomson Course Technology, Boston, 2005.

[SS07] M. Samer and S. Szeider. *Backdoor Sets of Quantified Boolean Formulas*. Proc. 10th Intl. Conf. on Theory and Applications of Satisfiability Testing (SAT 2007), Springer LNCS 4501, pp. 230–243, 2007.

[Sto76] L. Stockmeyer. *The Polynomial-Time Hierarchy*. Theoretical Computer Science, 3(1):1–22, 1976.

[Tse70] G. Tseitin. *On the Complexity of Derivation in Propositional Calculus*. In A. Silenko (ed.): Studies in Constructive Mathematics and Mathematical Logic, Part II, pp. 115–125, 1970.

[Wal70] W. Walkoe. *Finite Partially-Ordered Quantification*. Journal of Symbolic Logic, 35(4):535–555, 1970.

[Weg87] I. Wegener. *The Complexity of Boolean Functions*. Wiley-Teubner Series in Computer Science, B.G. Teubner, Stuttgart, 1987.

[Wra76] C. Wrathall. *Complete Sets and the Polynomial-Time Hierarchy*. Theoretical Computer Science, 3(1):23–33, 1976.

List of Figures

List of Listings

List of Tables

Index