Connectivity of Boolean Satisfiability

Von der

Fakultät für Elektrotechnik und Informatik
der Gottfried Wilhelm Leibniz Universität Hannover

zur Erlangung des Grades

Doktor der Naturwissenschaften
Dr. rer. nat.

genehmigte Dissertation

von

Dipl.-Physiker Konrad W. Schwerdtfeger

geboren am 6. August 1985 in Hildesheim

2016

Referent:	Heribert Vollmer, Leibniz Universität Hannover
Korrefferent:	Olaf Beyersdorff, University of Leeds
Tag der Promotion:	17.06.2016

Für meinen Vater
Gerhard Schwerdtfeger
1926 - 2014

Mein herzlicher Dank gilt meinem Doktorvater Heribert Vollmer
für seine Unterstützung bei der Arbeit an dieser Dissertation.

The first principle is that you must not fool yourself,
and you are the easiest person to fool.

Richard Feynman

Zusammenfassung

In dieser Dissertation befassen wir uns mit der Lösungsraum-Struktur Boolescher Erfüllbarkeits-Probleme, aus Sicht der theoretischen Informatik, insbesondere der Komplexitätstheorie.

Wir betrachten den *Lösungs-Graphen* Boolescher Formeln; dieser Graph hat als Knoten die Lösungen der Formel, und zwei Lösungen sind verbunden wenn sie sich in der Belegung genau einer Variablen unterscheiden. Für diesen implizit definierten Graphen untersuchen wir dann das Erreichbarkeitsproblem und das Zusammenhangsproblem.

Die erste systematische Untersuchung der Lösungs-Graphen Boolescher Constraint-Satisfaction-Probleme wurde 2006 von Gopalan et al. durchgeführt, motiviert hauptsächlich von Forschung für Erfüllbarkeits-Algorithmen. Insbesondere untersuchten sie $\mathrm{CNF_C}(\mathcal{S})$-Formeln, d.h. Konjunktionen von Bedingungen, welche sich aus dem Einsetzen von Variablen und Konstanten in Boolesche Relationen einer endlichen Menge $\mathcal{S}$ ergeben.

Gopalan et al. bewiesen eine Dichotomie für die Komplexität des Erreichbarkeitsproblems: Entweder ist es in Polynomialzeit lösbar oder PSPACE-vollständig, Damit übereinstimmend fanden sie auch eine strukturelle Dichotomie: Der maximale Durchmesser der Zusammenhangskomponenten ist entweder linear in der Zahl der Variablen, oder er kann exponentiell sein, Weiterhin vermuteten sie eine Trichotomie für das Zusammenhangsproblem: entweder ist es in P, coNP-vollständig oder PSPACE-vollständig. Zusammen mit Makino et al. bewiesen sie schon Teile dieser Trichotomie.

Auf diesen Arbeiten aufbauend vervollständigen wir hier den Beweis der Trichotomie, und korrigieren auch einen kleineren Fehler von Gopalan et al, was in einer leichten Verschiebung der Grenzen resultiert.

Anschließend untersuchen wir zwei wichtige Varianten: $\mathrm{CNF}(\mathcal{S})$-Formeln ohne Konstanten, und partiell quantifizierte Formeln. In beiden Fällen beweisen wir für das Erreichbarkeitsproblem und den Durchmesser Dichotomien analog jener für $\mathrm{CNF_C}(\mathcal{S})$-Formeln. Für das Zusammenhangsproblem zeigen wir eine Trichotomie im Fall quantifizierter Formeln, während wir im Fall der Formeln ohne Konstanten Fragmente identifizieren in denen das Problem in P, coNP-vollständig, und PSPACE-vollständig ist.

Schließlich betrachten wir die Zusammenhangs-Fragen für B-Formeln, d.h. geschachtelte Formeln, aufgebaut aus Junktoren einer endlichen Menge B, und für B-Circuits, d.h. Boolesche Schaltkreise, aufgebaut aus Gattern einer festen Menge B. Hier nutzen wir Emil Post's Klassifikation aller geschlossener Klassen Boolescher Funktionen. Wir beweisen eine gemeinsame Dichotomie für das Erreichbarkeitsproblem, das Zusammenhangsproblem und den Durchmesser: Auf der einen Seite sind beide Probleme in P und der Durchmesser ist linear, während auf der anderen Seite die Probleme PSPACE-vollständig sind und der Durchmesser exponentiell sein kann. Für partiell quantifizierte B-Formeln zeigen wir eine analoge Dichotomie.

Schlagworte Komplexität · Erfüllbarkeit · Zusammenhang in Graphen · Boolesche CSPs · Boolesche Schaltkreise · Post'scher Verband · Dichotomien

Abstract

In this thesis we are concerned with the solution-space structure of Boolean satisfiability problems, from the view of theoretical computer science, especially complexity theory.

We consider the *solution graph* of Boolean formulas; this is the graph where the vertices are the solutions of the formula, and two solutions are connected iff they differ in the assignment of exactly one variable. For this implicitly defined graph, we then study the *st*-connectivity and connectivity problems.

The first systematic study of the solution graphs of Boolean constraint satisfaction problems was done in 2006 by Gopalan et al., motivated mainly by research for satisfiability algorithms. In particular, they considered $\mathrm{CNF}_{\mathrm{C}}(\mathcal{S})$-formulas, which are conjunctions of constraints that arise from inserting variables and constants in relations of some finite set $\mathcal{S}$.

Gopalan et al. proved a computational dichotomy for the *st*-connectivity problem, asserting that it is either solvable in polynomial time or PSPACE-complete, and an aligned structural dichotomy, asserting that the maximal diameter of connected components is either linear in the number of variables, or can be exponential. Further, they conjectured a trichotomy for the connectivity problem: That it is either in P, coNP-complete, or PSPACE-complete. Together with Makino et al., they already proved parts of this trichotomy.

Building on this work, we here complete the proof of the trichotomy, and also correct a minor mistake of Gopalan et al., which leads to slight shifts of the boundaries.

We then investigate two important variants: $\mathrm{CNF}(\mathcal{S})$-formulas without constants, and partially quantified formulas. In both cases, we prove dichotomies for *st*-connectivity and the diameter analogous to the ones for $\mathrm{CNF}_{\mathrm{C}}(\mathcal{S})$-formulas. For for the connectivity problem, we show a trichotomy in the case of quantified formulas, while in the case of formulas without constants, we identify fragments where the problem is in P, where it is coNP-complete, and where it is PSPACE-complete.

Finally, we consider the connectivity issues for B-formulas, which are arbitrarily nested formulas built from some fixed set B of connectives, and for B-circuits, which are Boolean circuits where the gates are from some finite set B. Here, we make use of Emil Post's classification of all closed classes of Boolean functions. We prove a common dichotomy for both connectivity problems and the diameter: on one side, both problems are in P and the diameter is linear, while on the other, the problems are PSPACE-complete and the diameter can be exponential. For partially quantified B-formulas, we show an analogous dichotomy.

Keywords Computational complexity· Boolean satisfiability · Graph connectivity · Boolean CSPs · Boolean circuits · Post's lattice · Dichotomy theorems

Contents

1 Introduction **1**

1.1 Boolean Satisfiability and Solution Space Connectivity 1
1.2 Relevance of Solution Space Connectivity 2
1.3 Related Work, Prior Publications, and this Thesis 3
1.4 Associated Software 4
1.5 General Preliminaries 4

2 Connectivity of Constraints **7**

2.1 Preliminaries 7
2.1.1 CNF-Formulas and Schaefer's Framework 7
2.1.2 Classes of Relations 8
2.1.3 Classes of Sets of Relations 9
2.2 Results 11
2.3 The General Case: Reduction from a Turing Machine 12
2.4 Extension of PSPACE-Completeness: Structural Expressibility 12
2.5 Safely Tight Sets of Relations: Structure and Algorithms 20
2.6 CPSS Sets of Relations: A Simple Algorithm for Connectivity 22
2.7 The Last Piece: coNP-Hardness for Connectivity 24
2.7.1 Connectivity of Horn Formulas 25
2.7.2 Reduction from Satisfiability 27
2.7.3 Expressing M 28
2.8 Further Results about Constraint-Projection Separation 32

3 No-Constants and Quantified Variants **35**

3.1 No-Constants 35
3.1.1 st-Connectivity and Diameter 36
3.1.2 Deciding Connectivity via Constraint-Projection Separation . . 39
3.1.3 Deciding Connectivity via Self-Implication 41
3.1.4 coNP-Completeness for Connectivity within Schaefer 42
3.1.5 Reductions for Connectivity 46
3.2 Quantified Constraints 48
3.2.1 Properties that Persist 49
3.2.2 coNP-Completeness for Connectivity 50
3.2.3 Deciding Connectivity in Polynomial Time 52

4 Connectivity of Nested Formulas and Circuits **55**

4.1 Preliminaries: B-Circuits, B-Formulas, and Post's Lattice 55
4.2 Results 59
4.3 The Easy Side of the Dichotomy 59
4.4 The Hard Side of the Dichotomy 60
4.5 Quantified Formulas 65

5 Future Directions **69**

List of Figures

1.1.1 Depictions of the subgraph of the 5-dimensional hypercube graph induced by a typical random Boolean relation with 12 elements. 1
1.1.2 Subgraphs of the 8-dimensional hypercube graph induced by typical random relations 2

2.4.1 Proof of Step 1 of Lemma 2.4.6, and an example 15
2.7.1 An example for the proof of Lemma 2.7.9, illustrating the idea 28
2.7.2 A more complex example for the proof of Lemma 2.7.9 29

3.1.1 Producing a 1-isolating relation from every 3-ary relation R satisfying $110 \in R$ and $010 \notin R$ for the proof of Lemma 3.1.4 38

4.1.1 Post's lattice with our results 57
4.4.1 An example for the transformation in the proof of Lemma 4.4.5 63

List of Tables

2.1 Our classifications for $\mathrm{CNF_C}(\mathcal{S})$-formulas, in comparison to SAT 11

3.1 The classifications for $\mathrm{CNF}(\mathcal{S})$-formulas without constants 35
3.2 The classifications for Q-$\mathrm{CNF_C}(\mathcal{S})$-formulas 48

4.1 List of all closed classes of Boolean functions with definitions and bases 58

1 Introduction

1.1 Boolean Satisfiability and Solution Space Connectivity

The Boolean satisfiability problem (SAT) asks for a propositional formula if there is an assignment to the variables such that it evaluates to true. It is of great importance in many areas of theoretical and applied computer science: In complexity theory, it was one of the first problems proven to be NP-complete, and still is the most important standard problem for reductions. In propositional logic, many important reasoning problems can be reduced to SAT, e.g. checking entailment: For any two sentences α and β, $\alpha \models \beta$ if and only if $\alpha \wedge \overline{\beta}$ is unsatisfiable. These connections are used for example in artificial intelligence for reasoning, planning, and automated theorem proving, and in electronic design automation (EDA) for formal equivalence checking.

SAT is only the most basic version of a multitude of related problems, asking questions about a relation given by some short description. In one direction, one may look at constraint satisfaction problems over higher domains, or at multi-valued logics. In another direction, one may consider other tasks like enumerating all solutions, counting the solutions, checking the equivalence of formulas or circuits, or finding the optimal solution according to some measure. In this thesis, we follow the second direction and focus on the solution-space structure: For a formula ϕ, we consider the *solution graph* $G(\phi)$, where the vertices are the solutions, and two solutions are connected iff they differ in the assignment of exactly one variable. For this implicitly defined graph, we then study the connectivity and *st*-connectivity problems.

Since any propositional formula over n variables defines an n-ary Boolean relation R, i.e. a subset of $\{0,1\}^n$, another way to think of the solution graph is the subgraph of the n-dimensional hypercube graph induced by the vectors in R. The figures below give an impression of how solution graphs may look like.

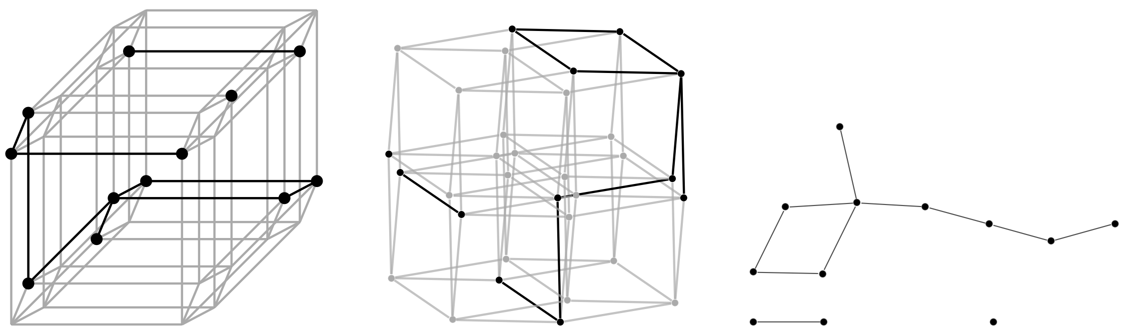


Figure 1.1.1 Depictions of the subgraph of the 5-dimensional hypercube graph induced by a typical random Boolean relation with 12 elements. Left: highlighted on an orthographic hypercube projection by our SatConn-tool. Center: highlighted on a "Spectral Embedding" of the hypercube graph by Mathematica. Right: the sole subgraph, arranged by Mathematica.

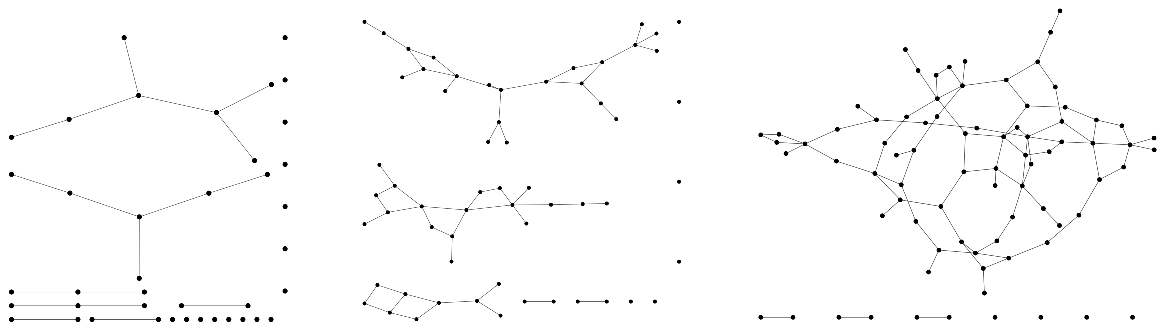


Figure 1.1.2 Subgraphs of the 8-dimensional hypercube graph (with 256 vertices) induced by typical random relations with 40, 60 and 80 elements, arranged by MATHEMATICA.

Our perspective is mainly from complexity theory: As it was done for SAT and many of the related problems, we classify restrictions of the connectivity problems by their worst-case complexity. Along the way, we will also examine structural properties of the solution graph, and devise efficient algorithms for solving the connectivity problems.

Besides the usual propositional formulas with the connectives $\wedge$, $\vee$ and $\neg$, there are many alternative representations of Boolean relations; we will consider the following:

- *Boolean constraint satisfaction problems* (*Boolean CSP*s, here *CSPs* for short), specifically
  - *$CNF_C(\mathcal{S})$-formulas*, i.e. conjunctions of constraints that arise from inserting variables and constants in relations of some finite set $\mathcal{S}$,
  - *$CNF(\mathcal{S})$-formulas*, where no constants may be used,
- *B-formulas*, i.e. arbitrarily nested formulas built from some finite set B of connectives,
- *B-circuits*, i.e. Boolean circuits where the gates are from some finite set B.

For $\mathrm{CNF_C}(\mathcal{S})$-formulas and B-formulas, we also consider versions with quantifiers.

1.2 Relevance of Solution Space Connectivity

A direct application of *st*-connectivity in solution graphs are *reconfiguration problems*, that arise when we wish to find a step-by-step transformation between two feasible solutions of a problem, such that all intermediate results are also feasible. Recently, the reconfiguration versions of many problems such as INDEPENDENT-SET, VERTEX-COVER, SET-COVER GRAPH-k-COLORING, SHORTEST-PATH have been studied (see e.g. [IDH+11, KMM11]).

The connectivity properties of solution graphs are also of relevance to the problem of *structure identification*, where one is given a relation explicitly and seeks a short representation of some kind (see e.g. [CKZ08]); this problem is important especially for learning in artificial intelligence.

Further, a better understanding of the solution space structure promises advancement of SAT algorithms: It has been discovered that the solution space connectivity is strongly correlated to the performance of standard satisfiability algorithms like Walk-SAT and DPLL on random instances: As one approaches the *satisfiability threshold* (the ratio of constraints to variables at which random k-CNF-formulas become unsatisfiable for $k \geq 3$) from below, the solution space (with the connectivity defined as above) fractures, and the performance of the algorithms deteriorates [MMZ05, MMW07]. These insights mainly came from statistical physics, and lead to the development of the *survey propagation algorithm*, which has superior performance on random instances [MMW07].

While current SAT solvers normally accept only CNF-formulas as input, in EDA the instances mostly derive from digital circuit descriptions [WLLH07], and although many such instances can easily be encoded in CNF, the original structural information, such as signal ordering, gate orientation and logic paths, is lost, or at least obscured. Since exactly this information can be very helpful for solving these instances, considerable effort has been made recently to develop satisfiability solvers that work with the circuit description directly [WLLH07], which have far better performance in EDA applications, or to restore the circuit structure from CNF [FM07]. This is a reason for us to study the solution space also for Boolean circuits.

1.3 Related Work, Prior Publications, and this Thesis

Research has focused on the solution space structure only quite recently: Complexity results for the connectivity problems in the solution graphs of CSPs have first been obtained in 2006 by P. Gopalan, P. G. Kolaitis, E. Maneva, and C. H. Papadimitriou [GKMP06, GKMP09]. In particular, they investigated $\mathrm{CNF_C}(\mathcal{S})$-formulas and studied

- the *st*-connectivity problem $\textsc{st-Conn}_\mathrm{C}(\mathcal{S})$, that asks for a $\mathrm{CNF_C}(\mathcal{S})$-formula ϕ and two solutions $\boldsymbol{s}$ and $\boldsymbol{t}$ whether there a path from $\boldsymbol{s}$ to $\boldsymbol{t}$ in $G(\phi)$,
- the connectivity problem $\textsc{Conn}_\mathrm{C}(\mathcal{S})$, that asks for a $\mathrm{CNF_C}(\mathcal{S})$-formula ϕ whether $G(\phi)$ is connected,

and

- the maximal diameter of any connected component of $G(\phi)$ for a $\mathrm{CNF_C}(\mathcal{S})$-formula ϕ, where the diameter of a component is the maximal shortest-path distance between any two vectors in that component.

They found a common structural and computational dichotomy: On one side, the maximal diameter is linear in the number of variables, *st*-connectivity is in P and connectivity is in coNP, while on the other side, the diameter can be exponential, and both problems are PSPACE-complete. Moreover, they conjectured a trichotomy for connectivity: That it is in P, coNP-complete, or PSPACE-complete. Together with Makino et al. [MTY07], they already proved parts of this trichotomy.

In [Sch13], we completed the proof of the trichotomy, and also corrected minor mistakes in [GKMP09], which lead to a slight shift of the boundaries towards the hard side. So for $\mathrm{CNF_C}(\mathcal{S})$-formulas, we now have a quite complete picture, which we present in Chapter 2. In [Sch13], we explained in detail the mistakes of Gopalan et al. and their implications, here we just give the correct statement and proofs.

In Chapter 3, we investigate two important variants: $\mathrm{CNF}(\mathcal{S})$-formulas without constants, and partially quantified $\mathrm{CNF_C}(\mathcal{S})$-formulas. In both cases, we prove a dichotomy for st-connectivity and the diameter analogous to the one for $\mathrm{CNF}_\mathrm{C}(\mathcal{S})$-formulas. For for the connectivity problem, we prove a trichotomy in the case of quantified formulas, while in the case of formulas without constants, we have no complete classification, but identify fragments where the problem is in P, where it is coNP-complete, and where it is PSPACE-complete. Of this chapter, only a preprint with preliminary results appeared on ArXiv [Sch14b].

Finally, in Chapter 4, we look at B-formulas and B-circuits. Here, we find a common dichotomy for the diameter and both connectivity problems: on one side, the diameter is linear and both problems are in P, while on the other, the diameter can be exponential, and the problems are PSPACE-complete. For quantified B-formulas, we prove an analogous dichotomy. The work in this chapter has been published in [Sch14a].

1.4 Associated Software

As part of the research for this thesis, several programs were written, some of which may be useful for future work on related problems. All software is written in Java (version 8) and provided in the SatConn package at https://github.com/konradws/SatConn, including a graphical tool to draw the solution graphs on hypercube projections, used for several graphics in this thesis.

After downloading the complete repository, the folder can be opened resp. imported in NetBeans or Eclipse as a project. The graphical tool is also provided as executable (SatConnTool.jar).

The most useful functions are declared `public` and equipped with Javadoc comments, where helpful. The `main`-functions provide usage examples and can be executed by running the respective file.

1.5 General Preliminaries

Prerequisites We assume familiarity with some basic concepts from theoretical computer science, especially complexity theory, and its mathematical foundations:

- From mathematics, we require propositional logic, and basics about graphs, hypergraphs, and lattices,
- From theoretical computer science, we require Turing machines, the common complexity classes P, NP, coNP, PSPACE, and polynomial-time reductions.

Notation We use $\boldsymbol{a}, \boldsymbol{b}, \ldots$ or $\boldsymbol{a}^1, \boldsymbol{a}^2, \ldots$ to denote vectors of Boolean values and $\boldsymbol{x}, \boldsymbol{y}, \ldots$ or $\boldsymbol{x}^1, \boldsymbol{x}^2, \ldots$ to denote vectors of variables, $\boldsymbol{a} = (a_1, a_2, \ldots)$ and $\boldsymbol{x} = (x_1, x_2, \ldots)$.

$\phi[\boldsymbol{x}^i/\boldsymbol{a}]$ denotes the formula resulting from ϕ by substituting the constants a_j for the variables x^i_j.

The symbol $\leq^p_m$ is used for polynomial-time many-one reductions.

Central concepts In the following definition, we formally introduce some concepts related to solution space connectivity in general. At the beginning of the next chapter, we define notions specific to CSPs. A reader only interested in B-formulas and B-circuits may read Section 2.3 after the next definition, and then skip to Chapter 4.

Definition 1.5.1 An n-ary *Boolean relation* (or *logical relation, relation* for short) is a subset of $\{0,1\}^n$ for some integer $n \geq 1$.

The set of solutions of a propositional formula ϕ over n variables defines in a natural way an n-ary relation $[\phi]$, where the variables are taken in lexicographic order. We will often identify the formula ϕ with the relation it defines and omit the brackets.

The *solution graph* $G(\phi)$ of ϕ then is the subgraph of the n-dimensional hypercube graph induced by the vectors in $[\phi]$. We will also refer to $G(R)$ for any logical relation R (not necessarily defined by a formula).

The *Hamming weight* $|\boldsymbol{a}|$ of a Boolean vector $\boldsymbol{a}$ is the number of 1's in $\boldsymbol{a}$. For two vectors $\boldsymbol{a}$ and $\boldsymbol{b}$, the *Hamming distance* $|\boldsymbol{a} - \boldsymbol{b}|$ is is the number of positions in which they differ.

If $\boldsymbol{a}$ and $\boldsymbol{b}$ are solutions of ϕ and lie in the same connected component (*component* for short) of $G(\phi)$, we write $d_\phi(\boldsymbol{a}, \boldsymbol{b})$ to denote the shortest-path distance between $\boldsymbol{a}$ and $\boldsymbol{b}$. The *diameter of a component* is the maximal shortest-path distance between any two vectors in that component. The *diameter of* $G(\phi)$ is the maximal diameter of any of its connected components.

2 Connectivity of Constraints

We start our investigation with constraint satisfaction problems. A constraint is a tuple of variables together with a Boolean relation, restricting the assignment of the variables. A CSP then is the question whether there is an assignment to all variables of a set of constraints such that all constraints are satisfied.

2.1 Preliminaries

2.1.1 CNF-Formulas and Schaefer's Framework

In line with Gopalan et al., we define CSPs by CNF($\mathcal{S}$)-formulas, which were introduced in 1978 by Thomas Schaefer as a generalization of CNF (conjunctive normal form) formulas [Sch78].

Definition 2.1.1 A *CNF-formula* is a propositional formula of the form $C_1 \wedge \cdots \wedge C_m$ ($1 \leq m < \infty$), where each C_i is a *clause*, that is, a finite disjunction of *literals* (variables or negated variables). A *k-CNF-formula* ($k \geq 1$) is a CNF-formula where each C_i has at most k literals. A *Horn (dual Horn)* formula is a CNF-formula where each C_i has at most one positive (negative) literal.

Definition 2.1.2 For a finite set of relations $\mathcal{S}$, a *CNF$_C$($\mathcal{S}$)-formula* over a set of variables V is a finite conjunction $C_1 \wedge \cdots \wedge C_m$, where each C_i is a *constraint application* (*constraint* for short), i.e., an expression of the form $R(\xi_1, \ldots, \xi_k)$, with a k-ary relation $R \in \mathcal{S}$, and each ξ_j is a variable from V or one of the constants $\{0, 1\}$. A *CNF($\mathcal{S}$)-formula* is a CNF$_\mathrm{C}$($\mathcal{S}$)-formula where each ξ_j is a variable in V, not a constant.

By $\mathrm{Var}(C_i)$, we denote the set of variables occurring in $\xi_1, \ldots, \xi_k$. With the *relation corresponding to C_i* we mean the relation $[R(\xi_1, \ldots, \xi_k)]$ (that may be different from R by substitution of constants, and identification and permutation of variables).

A *k-clause* is a disjunction of k variables or negated variables. For $0 \leq i \leq k$, let D_i be the corresponding to the k-clause whose first i literals are negated, and let $S_k = \{D_0, \ldots, D_k\}$, e.g., $\mathcal{S}_3 = \{[x \vee y \vee z], [\overline{x} \vee y \vee z], [\overline{x} \vee \overline{y} \vee z], [\overline{x} \vee \overline{y} \vee \overline{z}]\}$. Thus, CNF($S_k$) is the collection of k-CNF-formulas.

Thomas Schaefer introduced CNF($\mathcal{S}$)-formulas for expressing variants of Boolean satisfiability; in his dichotomy theorem, Schaefer then classified the complexity of the satisfiability problem for CNF$_\mathrm{C}$($\mathcal{S}$)- and CNF($\mathcal{S}$)-formulas [Sch78]; we will do so here for the connectivity problems. We use the following notation:

- SAT($\mathcal{S}$) for the *satisfiability problem:* Given a CNF($\mathcal{S}$)-formula ϕ, is ϕ satisfiable?
- ST-CONN($\mathcal{S}$) for the *st-connectivity problem*: Given a CNF($\mathcal{S}$)-formula ϕ and two solutions $\boldsymbol{s}$ and $\boldsymbol{t}$, is there a path from $\boldsymbol{s}$ to $\boldsymbol{t}$ in $G(\phi)$?
- CONN($\mathcal{S}$) for the *connectivity problem:* Given a CNF($\mathcal{S}$)-formula ϕ, is $G(\phi)$ connected? (if ϕ is unsatisfiable, we consider $G(\phi)$ connected)

The respective problems for $\mathrm{CNF_C}(\mathcal{S})$-formulas are marked with the subscript $_\mathrm{C}$. Note that Gopalan et al. considered the case with constants, but omitted the $_\mathrm{C}$.

2.1.2 Classes of Relations

In the following definition, we introduce the types of relations needed for the classifications. Some are already familiar from Schaefer's dichotomy theorem, some were introduced by Gopalan et al., and the ones starting with "safely" we defined in [Sch13] to account for the shift of the boundaries resulting from Gopalan et al.'s mistake; *IHSB* stands for "implicative hitting set-bounded" and was introduced in [CKS01].

Definition 2.1.3 Let R be an n-ary logical relation.

- R is *0-valid (1-valid)* if $0^n \in R$ ($1^n \in R$).
- R is *complementive* if for every vector $(a_1, \ldots, a_n) \in R$, also $(a_1 \oplus 1, \ldots, a_n \oplus 1) \in R$.
- R is *bijunctive* if it is the set of solutions of a 2-CNF-formula.
- R is *Horn (dual Horn)* if it is the set of solutions of a Horn (dual Horn) formula.
- R is *affine* if it is the set of solutions of a formula $x_{i_1} \oplus \ldots \oplus x_{i_m} \oplus c$ with $i_1, \ldots, i_m \in \{1, \ldots, n\}$ and $c \in \{0, 1\}$.
- R is *componentwise bijunctive* if every connected component of $G(R)$ is a bijunctive relation. R is *safely componentwise bijunctive* if R and every relation R' obtained from R by identification of variables is componentwise bijunctive.
- R is *OR-free* (*NAND-free*) if the relation $\mathrm{OR} = \{01, 10, 11\}$ ($\mathrm{NAND} = \{00, 01, 10\}$) cannot be obtained from R by substitution of constants. R is *safely OR-free* (*safely NAND-free*) if R and every relation R' obtained from R by identification of variables is OR-free (NAND-free).
- R is *IHSB− (IHSB+)* if it is the set of solutions of a Horn (dual Horn) formula in which all clauses with more than 2 literals have only negative literals (only positive literals).
- R is *componentwise IHSB− (componentwise IHSB+)* if every connected component of $G(R)$ is IHSB− (IHSB+). R is *safely componentwise IHSB− (safely componentwise IHSB+)* if R and every relation R' obtained from R by identification of variables is componentwise IHSB− (componentwise IHSB+).

If one is given the relation explicitly (as a set of vectors), the properties 0-valid, 1-valid, complementive, OR-free and NAND-free can be checked straightforward, while bijunctive, Horn, dual Horn, affine, IHSB− and IHSB+ can be checked by *closure properties*:

Definition 2.1.4 A relation R is *closed* under some n-ary operation f iff the vector obtained by the coordinate-wise application of f to any m vectors from R is again in R, i.e., if

$$\boldsymbol{a}^1, \ldots, \boldsymbol{a}^m \in R \Longrightarrow \big(f(a_1^1, \ldots, a_1^m), \ldots, f(a_n^1, \ldots, a_n^m)\big) \in R.$$

Lemma 2.1.5 *A relation R is*

1. *bijunctive, iff it is closed under the ternary majority operation* $maj(x, y, z){=}(x \vee y) \wedge (y \vee z) \wedge (z \vee x)$,
2. *Horn (dual Horn), iff it is closed under $\wedge$ (under $\vee$, resp.),*
3. *affine, iff it is closed under $x \oplus y \oplus z$,*
4. *IHSB− (IHSB+), iff it is closed under $x \wedge (y \vee z)$ (under $x \vee (y \wedge z)$, resp.).*

Proof. 1. See [CKS01, Lemma 4.9].
2. See [CKS01, Lemma 4.8].
3. See [CKS01, Lemma 4.10].
4. This can be verified using the Galois correspondence between closed sets of relations and closed sets of Boolean functions (see [BRSV05]): From the table (Fig. 1) in [BRSV05], we find that the IHSB− relations are a base of the co-clone $\mathrm{INV}(\mathsf{S}_{10})$, and the IHSB+ ones a base of $\mathrm{INV}(\mathsf{S}_{00})$, and from the table (Figure 1) in [BCRV03], we see that $x \wedge (y \vee z)$ and $x \vee (y \wedge z)$ are bases of the clones S_{10} and S_{00}, resp. □

Remark 2.1.6. The class `Check` of SatConn provides functions to check the properties of Definition 2.1.3, and the class `Clones` provides functions to calculate the clone and co-clone of a relation.

The closure properties carry over from a relation to its connected components, as shown by Gopalan et al.:

Lemma 2.1.7 [GKMP09, Lemma 4.1] *If a logical relation R is closed under an operation $\alpha : \{0,1\}^k \to \{0,1\}$ such that $\alpha(1, \dots, 1) = 1$ and $\alpha(0, \dots, 0) = 0$ (a.k.a. an idempotent operation), then every connected component of $G(R)$ is closed under α.*

2.1.3 Classes of Sets of Relations

The classes in the following definition demarcate the structural and computational boundaries for the solution graphs of $\mathrm{CNF_C}(\mathcal{S})$-formulas.

Definition 2.1.8 A set $\mathcal{S}$ of logical relations is *safely tight* if at least one of the following conditions holds:

1. Every relation in $\mathcal{S}$ is safely componentwise bijunctive.
2. Every relation in $\mathcal{S}$ is safely OR-free .
3. Every relation in $\mathcal{S}$ is safely NAND-free.

A set $\mathcal{S}$ of logical relations is *Schaefer* if at least one of the following conditions holds:

1. Every relation in $\mathcal{S}$ is bijunctive.
2. Every relation in $\mathcal{S}$ is Horn.
3. Every relation in $\mathcal{S}$ is dual Horn.
4. Every relation in $\mathcal{S}$ is affine.

A set $\mathcal{S}$ of logical relations is *CPSS* if at least one of the following conditions holds:

1. Every relation in $\mathcal{S}$ is bijunctive.
2. Every relation in $\mathcal{S}$ is Horn and safely componentwise IHSB$-$.
3. Every relation in $\mathcal{S}$ is dual Horn and safely componentwise IHSB$+$.
4. Every relation in $\mathcal{S}$ is affine.

A single logical relation R is safely tight, Schaefer, or CPSS, if $\{R\}$ has that property. Vice versa, we say that a set $\mathcal{S}$ of logical relations has one of the properties from Definition 2.1.3 if every relation in $\mathcal{S}$ has that property, e.g., $\mathcal{S}$ is 0-valid if every relation in $\mathcal{S}$ is 0-valid.

The term *tight* was introduced by Gopalan et al. because of the structural properties of the formulas built from tight (actually, only safely tight) relations, see Lemma 2.5.1 and Lemma 2.5.4. We introduced the CPSS class in [Sch13]; CPSS stands for "constraint-projection separating Schaefer", which will become clear in Section 2.6 from Definition 2.6.1, Lemma 2.6.4 and Lemma 2.8.1.

From the definition we see that every CPSS set of relations is also Schaefer, and we can show that it also holds that every Schaefer set is safely tight, by modifying a lemma of Gopalan et al.:

Lemma 2.1.9 [modified from GKMP09, Lemma 4.2] *Let R be a logical relation.*

1. *If R is bijunctive, then it is safely componentwise bijunctive.*
2. *If R is Horn, then it is safely OR-free.*
3. *If R is dual Horn, then it is safely NAND-free.*
4. *If R is affine, then it is safely componentwise bijunctive, safely OR-free, and safely NAND-free.*

Proof. We first note that

(*) any relation obtained from a bijunctive (Horn, dual Horn, affine) one by identification of variables is itself bijunctive (Horn, dual Horn, affine),

which is obvious from the definitions.

If R is bijunctive, it is closed under maj, which is idempotent, so by Lemma 2.1.7, R is also componentwise bijunctive, and by (*), it is safely componentwise bijunctive as well.

The cases of Horn and dual Horn are symmetric. Suppose a r-ary Horn relation R is not OR-free. Then there exist $i, j \in \{1, \ldots, r\}$ and constants $t_1, \ldots, t_r \in \{0, 1\}$ such that the relation $R(t_1, \ldots, t_{i-1}, x, t_{i+1}, \ldots, t_{j-1}, y, t_{j+1}, \ldots, t_r)$ on variables x and y is equivalent to $x \vee y$, i.e.

$$R(t_1, \ldots, t_{i-1}, x, t_{i+1}, \ldots, t_{j-1}, y, t_{j+1}, \ldots, t_r) = \{01, 11, 10\}.$$

Thus the tuples $\boldsymbol{t^{00}}, \boldsymbol{t^{01}} \boldsymbol{t^{10}}, \boldsymbol{t^{11}}$ defined by $(t_i^{ab}, t_j^{ab}) = (a, b)$ and $t_k^{ab} = t_k$ for every $k \notin \{i, j\}$, where $a, b, \in \{0, 1\}$ satisfy $\boldsymbol{t^{10}}, \boldsymbol{t^{11}}, \boldsymbol{t^{01}} \in R$ and $\boldsymbol{t^{00}} \notin R$. However, since every Horn relation is closed under $\wedge$, it follows that $\boldsymbol{t}^{01} \wedge \boldsymbol{t}^{10} = \boldsymbol{t}^{00}$ must be in R, which is a contradiction. So R is OR-free, and again by (*), it must be safely OR-free as well.

For the affine case, a small modification of the last step of the above argument shows that an affine relation also is OR-free; therefore, dually, it is also NAND-free. Namely, since a relation R is affine if and only if it is closed under ternary $\oplus$, it follows that $\boldsymbol{t}^{01} \oplus \boldsymbol{t}^{11} \oplus \boldsymbol{t}^{10} = \boldsymbol{t}^{00}$ must be in R. Since the connected components of an affine relation are both OR-free and NAND-free the subgraphs that they induce are hypercubes, which are also bijunctive relations. Therefore an affine relation is also componentwise bijunctive. With this, it must also be safely OR-free, safely OR-free and safely componentwise bijunctive by (*). $\square$

2.2 Results

We are now ready to state the results for $\mathrm{CNF}_{\mathrm{C}}(\mathcal{S})$-formulas; in the subsequent sections we will prove them. The following two theorems give complete classifications up to polynomial-time isomorphisms. They are summarized in the table below.

<table>
<tr><th>$\mathcal{S}$</th><th>$\textsc{Sat}_{\mathrm{C}}(\mathcal{S})$</th><th>$\textsc{Conn}_{\mathrm{C}}(\mathcal{S})$</th><th>$\textsc{st-Conn}_{\mathrm{C}}(\mathcal{S})$</th><th>Diameter</th></tr>
<tr><td>not safely tight</td><td rowspan="2">NP-complete</td><td>PSPACE-complete</td><td>PSPACE-complete</td><td>$2^{\Omega(\sqrt{n})}$</td></tr>
<tr><td>safely tight, not Schaefer</td><td rowspan="2">coNP-complete</td><td rowspan="3">in P</td><td rowspan="3">$O(n)$</td></tr>
<tr><td>Schaefer, not CPSS</td><td rowspan="2">in P</td></tr>
<tr><td>CPSS</td><td>in P</td></tr>
</table>

Table 2.1 Our classifications for $\mathrm{CNF}_{\mathrm{C}}(\mathcal{S})$-formulas, in comparison to $\textsc{Sat}$.

Theorem 2.2.1 (Dichotomy theorem for $\textsc{st-Conn}_{\mathrm{C}}(\mathcal{S})$ and the diameter) *Let $\mathcal{S}$ be a finite set of logical relations.*

1. *If $\mathcal{S}$ is safely tight, $\textsc{st-Conn}_{\mathrm{C}}(\mathcal{S})$ is in* P*, and for every $\mathrm{CNF}_{\mathrm{C}}(\mathcal{S})$-formula ϕ, the diameter of $G(\phi)$ is linear in the number of variables.*
2. *Otherwise, $\textsc{st-Conn}_{\mathrm{C}}(\mathcal{S})$ is* PSPACE*-complete, and there are $\mathrm{CNF}_{\mathrm{C}}(\mathcal{S})$-formulas ϕ, such that the diameter of $G(\phi)$ is exponential in the number of variables.*

Proof. 1. See Lemma 2.5.6.
2. See Corollary 2.4.9. $\square$

Theorem 2.2.2 (Trichotomy theorem for $\textsc{Conn}_{\mathrm{C}}(\mathcal{S})$) *Let $\mathcal{S}$ be a finite set of logical relations.*

1. *If $\mathcal{S}$ is CPSS, $\textsc{Conn}_{\mathrm{C}}(\mathcal{S})$ is in* P.
2. *Else if $\mathcal{S}$ is safely tight, $\textsc{Conn}_{\mathrm{C}}(\mathcal{S})$ is* coNP*-complete.*
3. *Else, $\textsc{Conn}_{\mathrm{C}}(\mathcal{S})$ is* PSPACE*-complete.*

Proof. 1. See Corollary 2.6.6.
2. See Corollary 2.7.11.
3. See Corollary 2.4.9. $\square$

2.3 The General Case: Reduction from a Turing Machine

We start with the general case. Gopalan et al. showed that for 3-CNF-formulas, st-Conn$_C$ and Conn$_C$ are PSPACE-complete, and the diameter can be exponential:

Lemma 2.3.1 [GKMP09, Lemma 3.6] *For general CNF-formulas, as well as for 3-CNF-formulas,* st-Conn$_C$ *and* Conn$_C$ *are* PSPACE*-complete.*

Showing that the problems are in PSPACE is straightforward: Given a CNF-formula ϕ and two solutions $\boldsymbol{s}$ and $\boldsymbol{t}$, we can guess a path of length at most 2^n between them and verify that each vertex along the path is indeed a solution. Hence st-Conn is in NPSPACE, which equals PSPACE by Savitch's theorem. For Conn, by reusing space we can check for all pairs of vectors whether they are satisfying, and, if they both are, whether they are connected in $G(\phi)$.

The hardness-proof is quite intricate: it consists of a direct reduction from the computation of a space-bounded Turing machine M. The input-string w of M is mapped to a 3-CNF-formula ϕ and two satisfying assignments $\boldsymbol{s}$ and $\boldsymbol{t}$, corresponding to the initial and accepting configuration of a Turing machine M' constructed from M and w, s.t. $\boldsymbol{s}$ and $\boldsymbol{t}$ are connected in $G(\phi)$ iff M accepts w. Further, all satisfying assignments of ϕ are connected to either $\boldsymbol{s}$ or $\boldsymbol{t}$, so that $G(\phi)$ is connected iff M accepts.

Lemma 2.3.2 [GKMP09, Lemma 3.7] *For n even, there is a 3-CNF-formula ϕ_n with n variables and $O(n^2)$ clauses, s.t. $G(\phi_n)$ is a path of length greater than $2^{\frac{n}{2}}$.*

The proof of this lemma is by direct construction of such a formula.

2.4 Extension of PSPACE-Completeness: Structural Expressibility

To show that PSPACE-hardness and exponential diameter extend to all not (safely) tight sets of relations, Gopalan et al. used the concept of structural expressibility, which is a modification of Schaefer's "representability" that he used for his dichotomy theorem[1], so let us have a quick look at this first:

Theorem 2.4.1 (Schaefer's dichotomy theorem [Sch78]) *Let $\mathcal{S}$ be a finite set of logical relations.*

1. *If $\mathcal{S}$ is Schaefer, then* Sat$_C(\mathcal{S})$ *is in P; otherwise,* Sat$_C(\mathcal{S})$ *is* NP*-complete.*
2. *If $\mathcal{S}$ is 0-valid, 1-valid, or Schaefer, then* Sat$(\mathcal{S})$ *is in P; otherwise,* Sat$(\mathcal{S})$ *is* NP*-complete.*[2]

[1] While Schaefer's dichotomy theorem and many related complexity classifications can also be proved using Post's classification of all closed classes of Boolean functions and a Galois correspondence (see e.g. [CKV08]), this seems not possible for our connectivity problems: The boundaries here "cut across Boolean clones" (more exactly: co-clones), as already Gopalan et al. noted [GKMP09]. For example, the co-clone of both $R = \{100, 010, 001\}$ and $R' = \{100, 010, 001, 110, 101\}$ is I_2, but R is safely OR-free and thus tight, while R' is not safely tight.

[2] Here we assume that $\mathcal{S}$ contains no empty relations, see Section 3.1.

Schaefer first proved statement 1, and from that derived the no-constants version; we here discuss only the proof statement 1.

Schaefer used a reduction from satisfiability of 3-CNF-formulas, i.e. $\mathrm{CNF_C}(\mathcal{S}_3)$-formulas (see Definition 2.1.2), which was already known to be NP-complete by the Cook–Levin theorem. Therefor, he exploited that any existentially quantified formula is satisfiability-equivalent to the formula with the quantifiers removed, and introduced the notion of *representability*, that became also know as *expressibility*:

Definition 2.4.2 A relation R is *expressible* from a set of relations $\mathcal{S}$ if there is a $\mathrm{CNF_C}(\mathcal{S})$-formula $\phi(\boldsymbol{x}, \boldsymbol{y})$ such that $R = \{\boldsymbol{a} | \exists \boldsymbol{y} \phi(\boldsymbol{a}, \boldsymbol{y})\}$.

He then showed that every Boolean relation is expressible from any set of relations that is not Schaefer, and that this expression can efficiently be constructed.

With this, it is easy to see that for every non-Schaefer set $\mathcal{S}$, satisfiability of any $\mathrm{CNF_C}(\mathcal{S}_3)$-formula ψ can be reduced to satisfiability of a $\mathrm{CNF_C}(\mathcal{S})$-formula, constructed as follows: Replace in ψ every constraint $R(\boldsymbol{\xi})$ by $\phi(\boldsymbol{\xi}, \boldsymbol{y})$ with ϕ from Definition 2.4.2, and new variables $\boldsymbol{y}$, distinct for each constraint.

As Gopalan et al. explain in section 3.1 of [GKMP09], for the connectivity problems, expressibility is not sufficient; therefore, they introduced *structural expressibility*:

Definition 2.4.3 A relation R is *structurally expressible* from a set of relations $\mathcal{S}$ if there is a $\mathrm{CNF_C}(\mathcal{S})$-formula ϕ such that the following conditions hold:

1. $R = \{\boldsymbol{a} | \exists \boldsymbol{y} \phi(\boldsymbol{a}, \boldsymbol{y})\}$.
2. For every $\boldsymbol{a} \in R$, the graph $G(\phi(\boldsymbol{a}, \boldsymbol{y}))$ is connected.
3. For $\boldsymbol{a}, \boldsymbol{b} \in R$ with $|\boldsymbol{a} - \boldsymbol{b}| = 1$, there exists a *witness* $\boldsymbol{w}$ such that $(\boldsymbol{a}, \boldsymbol{w})$ and $(\boldsymbol{b}, \boldsymbol{w})$ are solutions of ϕ.

Gopalan et al. now argued that connectivity were retained when replacing every constraint R with a structural expression of R in a $\mathrm{CNF_C}(\mathcal{S})$-formula. In fact, this is only true for $\mathrm{CNF_C}(\mathcal{S})$-formulas where no variable is used more than once in any constraint, and their proof is only correct for such formulas that also use no constants:

Lemma 2.4.4 [corrected from GKMP09, Lemma 3.2] *Let $\mathcal{S}$ and $\mathcal{S}'$ be sets of relations such that every $R \in \mathcal{S}'$ is structurally expressible from $\mathcal{S}$. Given a CNF($\mathcal{S}'$)-formula $\psi(\boldsymbol{x})$ (without constants), where no variable is used more than once in any constraint, one can efficiently construct a CNF_C($\mathcal{S}$)-formula $\varphi(\boldsymbol{x}, \boldsymbol{y})$ such that*

1. *$\psi(\boldsymbol{x}) = \exists \boldsymbol{y} \varphi(\boldsymbol{x}, \boldsymbol{y})$;*
2. *if $(\boldsymbol{s}, \boldsymbol{w}^s)$, $(\boldsymbol{t}, \boldsymbol{w}^t)$ are connected in $G(\varphi)$ by a path of length d, then there is a path from $\boldsymbol{s}$ to $\boldsymbol{t}$ in $G(\psi)$ of length at most d;*
3. *if $\boldsymbol{s}, \boldsymbol{t} \in \psi$ are connected in $G(\psi)$, then for every witness $\boldsymbol{w}^s$ of $\boldsymbol{s}$, and every witness $\boldsymbol{w}^t$ of $\boldsymbol{t}$, there is a path from $(\boldsymbol{s}, \boldsymbol{w}^s)$ to $(\boldsymbol{t}, \boldsymbol{w}^t)$ in $G(\varphi)$.*

In Gopalan et al.'s proof, we only clarify the notation a little:

Proof. Let $\psi(\boldsymbol{x}) = C_1 \wedge \cdots \wedge C_m$ with $C_j = R_j(\boldsymbol{x}_j)$, where R_j is some relation from $\mathcal{S}'$, and $\boldsymbol{x}_j$ is the vector of variables to which R_j is applied. Let φ_j be the structural expression for R_j from $\mathcal{S}$, so that $R_j(\boldsymbol{x}_j) \equiv \exists \boldsymbol{y}_j\ \varphi_j(\boldsymbol{x}_j, \boldsymbol{y}_j)$. Let $\boldsymbol{y}$ be the vector $(\boldsymbol{y}_1, \ldots, \boldsymbol{y}_m)$ and let $\varphi(\boldsymbol{x}, \boldsymbol{y})$ be the formula $\wedge_{j=1}^m \varphi_j(\boldsymbol{x}_j, \boldsymbol{y}_j)$. Then $\psi(\boldsymbol{x}) \equiv \exists \boldsymbol{y}\ \varphi(\boldsymbol{x}, \boldsymbol{y})$.

Statement 2 follows from 1 by projection of the path on the coordinates of $\boldsymbol{x}$. For statement 3, consider $\boldsymbol{s}, \boldsymbol{t} \in \psi$ that are connected in $G(\psi)$ via a path $\boldsymbol{s} = \boldsymbol{u^0} \to \boldsymbol{u^1} \to \cdots \to \boldsymbol{u^r} = \boldsymbol{t}$. For every $\boldsymbol{u^i}, \boldsymbol{u^{i+1}}$, and clause C_j, there exists an assignment $\boldsymbol{w^i_j}$ to $\boldsymbol{y}_j$ such that both $(\boldsymbol{u^i_j}, \boldsymbol{w^i_j})$ and $(\boldsymbol{u^{i+1}_j}, \boldsymbol{w^i_j})$ are solutions of φ_j, by condition 3 of structural expressibility. Thus $(\boldsymbol{u^i}, \boldsymbol{w^i})$ and $(\boldsymbol{u^{i+1}}, \boldsymbol{w^i})$ are both solutions of φ, where $\boldsymbol{w^i} = (\boldsymbol{w^i_1}, \ldots, \boldsymbol{w^i_m})$. Further, for every $\boldsymbol{u^i}$, the space of solutions of $\varphi(\boldsymbol{u^i}, \boldsymbol{y})$ is the product space of the solutions of $\varphi_j(\boldsymbol{u^i_j}, \boldsymbol{y}_j)$ over $j = 1, \ldots, m$. Since these are all connected by condition 2 of structural expressibility, $G(\varphi(\boldsymbol{u^i}, \boldsymbol{y}))$ is connected. The following describes a path from $(\boldsymbol{s}, \boldsymbol{w^s})$ to $(\boldsymbol{t}, \boldsymbol{w^t})$ in $G(\varphi)$: $(\boldsymbol{s}, \boldsymbol{w^s}) \rightsquigarrow (\boldsymbol{s}, \boldsymbol{w^0}) \to (\boldsymbol{u^1}, \boldsymbol{w^0}) \rightsquigarrow (\boldsymbol{u^1}, \boldsymbol{w^1}) \to \cdots \rightsquigarrow (\boldsymbol{u^{r-1}}, \boldsymbol{w^{r-1}}) \to (\boldsymbol{t}, \boldsymbol{w^{r-1}}) \rightsquigarrow (\boldsymbol{t}, \boldsymbol{w^t})$. Here $\rightsquigarrow$ indicates a path in $G(\varphi(\boldsymbol{u^i}, \boldsymbol{y}))$. □

It is easy to show that the statement of Lemma 2.4.4 is also correct if we allow constants in ψ; however, we don't need this result. In [Sch13], we explain in detail the problem with repeated variables in constraint applications.

We have to change Gopalan et al.'s corollary accordingly; we denote the connectivity problems for CNF($\mathcal{S}$)-formulas without repeated variables in constraints (and without constants) by the subscript $_{\mathrm{ni}}$:

Corollary 2.4.5 [corrected from GKMP09, Corollary 3.3] *Suppose $\mathcal{S}$ and $\mathcal{S}'$ are sets of relations such that every $R \in \mathcal{S}'$ is structurally expressible from $\mathcal{S}$.*

1. *There are polynomial-time reductions from* $\textsc{Conn}_{\mathrm{ni}}(\mathcal{S}')$ *to* $\textsc{Conn}_{\mathrm{C}}(\mathcal{S})$*, and from* st-$\textsc{Conn}_{\mathrm{ni}}(\mathcal{S}')$ *to* st-$\textsc{Conn}_{\mathrm{C}}(\mathcal{S})$.
2. *If there exists a* $\mathrm{CNF}_{\mathrm{ni}}(\mathcal{S}')$*-formula $\psi(\boldsymbol{x})$ with n variables, m clauses and diameter d, then there exists a* $\mathrm{CNF}_{\mathrm{C}}(\mathcal{S})$*-formula $\phi(\boldsymbol{x}, \boldsymbol{y})$, where $\boldsymbol{y}$ is a vector of $O(m)$ variables, such that the diameter of $G(\phi)$ is at least d.*

Since 3-CNF-fomulas are $\mathrm{CNF}_{\mathrm{ni}}(\mathcal{S}_1 \cup \mathcal{S}_2 \cup \mathcal{S}_3)$-formulas, for the reductions to work it now remains to prove that $\mathcal{S}_1 \cup \mathcal{S}_2 \cup \mathcal{S}_3$ is structurally expressible from any not safely tight set. As Theorem 2.4.8 below shows, in fact every Boolean relation is structurally expressible from any such set. The long proof of the next lemma contains only minor modifications from [GKMP09].

Lemma 2.4.6 [modified from GKMP09, Lemma 3.4] *If a set $\mathcal{S}$ of relations is not safely tight, $\mathcal{S}_3$ is structurally expressible from $\mathcal{S}$.*

Proof. First, observe that all 2-clauses are structurally expressible from $\mathcal{S}$. There exists $R \in \mathcal{S}$ which is not safely OR-free, so we can express $(x_1 \vee x_2)$ by substituting constants and identifying variables in R. Similarly, we can express $(\bar{x}_1 \vee \bar{x}_2)$ using a relation that is not safely NAND-free. The last 2-clause $(x_1 \vee \bar{x}_2)$ can be obtained from OR and NAND by a technique that corresponds to reverse resolution. $(x_1 \vee \bar{x}_2) = \exists y\ (x_1 \vee y) \wedge (\bar{y} \vee \bar{x}_2)$. It is easy to see that this gives a structural expression. From here onwards we assume that $\mathcal{S}$ contains all 2-clauses. The proof now proceeds in four steps. First, we will express a relation in which there exist two elements that are at graph distance larger than their Hamming distance. Second, we will express a relation that is just a single path between such elements. Third, we will express a relation which is a path of length 4 between elements at Hamming distance 2. Finally, we will express the 3-clauses.

Step 1. Structurally expressing a relation in which some distance expands. For a relation R, we say that the distance between $\boldsymbol{a}$ and $\boldsymbol{b}$ *expands* if $\boldsymbol{a}$ and $\boldsymbol{b}$ are connected in $G(R)$, but $d_R(\boldsymbol{a}, \boldsymbol{b}) > |\boldsymbol{a} - \boldsymbol{b}|$. Later on, we will show that no distance expands in safely componentwise bijunctive relations. The same also holds true for the relation $R_{\mathrm{NAE}} = \{0,1\}^3 \setminus \{000, 111\}$, which is not safely componentwise bijunctive. Nonetheless, we show here that if R is not safely componentwise bijunctive, then, by adding 2-clauses, we can structurally express a relation Q in which some distance expands. For instance, when $R = R_{\mathrm{NAE}}$, then we can take $Q(x_1, x_2, x_3) = R_{\mathrm{NAE}}(x_1, x_2, x_3) \wedge (\bar{x_1} \vee \bar{x}_3)$. The distance between $\boldsymbol{a} = 100$ and $\boldsymbol{b} = 001$ in Q expands. Similarly, in the general construction, we identify $\boldsymbol{a}$ and $\boldsymbol{b}$ on a cycle, and add 2-clauses that eliminate all the vertices along the shorter arc between $\boldsymbol{a}$ and $\boldsymbol{b}$.

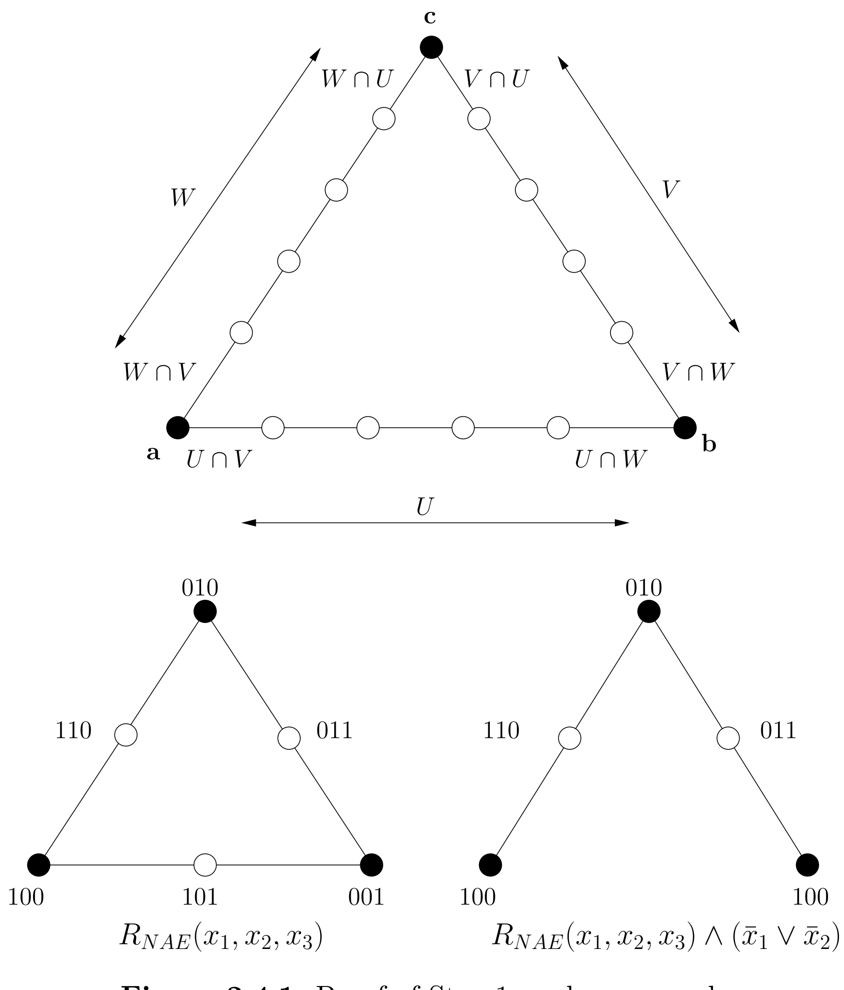


Figure 2.4.1 Proof of Step 1, and an example.

Since $\mathcal{S}$ is not safely tight, it contains a relation which is not safely componentwise bijunctive, from which we can obtain a not componentwise bijunctive relation R. If R contains $\boldsymbol{a}, \boldsymbol{b}$ where the distance between them expands, we are done. So assume that for all $\boldsymbol{a}, \boldsymbol{b} \in G(R)$, $d_R(\boldsymbol{a}, \boldsymbol{b}) = |\boldsymbol{a} - \boldsymbol{b}|$. Since R is not componentwise bijunctive, there exists a triple of assignments $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}$ lying in the same component such that $\mathrm{Maj}(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c})$ is not in that component (which also easily implies it is not in R). Choose the triple such that the sum of pairwise distances $d_R(\boldsymbol{a}, \boldsymbol{b}) + d_R(\boldsymbol{b}, \boldsymbol{c}) + d_R(\boldsymbol{c}, \boldsymbol{a})$ is minimized. Let $U = \{i | a_i \neq b_i\}$, $V = \{i | b_i \neq c_i\}$, and $W = \{i | c_i \neq a_i\}$. Since $d_R(\boldsymbol{a}, \boldsymbol{b}) = |\boldsymbol{a} - \boldsymbol{b}|$, a shortest path does not flip variables outside of U, and each variable in U is flipped exactly once. The same holds for V and W. We note some useful properties of the sets U, V, W.

1. *Every index $i \in U \cup V \cup W$ occurs in exactly two of U, V, W.*
   Consider going by a shortest path from $\boldsymbol{a}$ to $\boldsymbol{b}$ to $\boldsymbol{c}$ and back to $\boldsymbol{a}$. Every $i \in U \cup V \cup W$ is seen an even number of times along this path since we return to $\boldsymbol{a}$. It is seen at least once, and at most thrice, so in fact it occurs twice.
2. *Every pairwise intersection $U \cap V, V \cap W$ and $W \cap U$ is non-empty.*
   Suppose the sets U and V are disjoint. From Property 1, we must have $W = U \cup V$. But then it is easy to see that $\mathrm{Maj}(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}) = \boldsymbol{b}$ which is in R. This contradicts the choice of $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}$.
3. *The sets $U \cap V$ and $U \cap W$ partition the set U.*
   By Property 1, each index of U occurs in one of V and W as well. Also since no index occurs in all three sets U, V, W this is in fact a disjoint partition.
4. *For each index $i \in U \cap W$, it holds that $\boldsymbol{a} \oplus \boldsymbol{e}_i \notin R$.*
   Assume for the sake of contradiction that $\boldsymbol{a'} = \boldsymbol{a} \oplus \boldsymbol{e}_i \in R$. Since $i \in U \cap W$ we have simultaneously moved closer to both $\boldsymbol{b}$ and $\boldsymbol{c}$. Hence we have $d_R(\boldsymbol{a'}, \boldsymbol{b}) + d_R(\boldsymbol{b}, \boldsymbol{c}) + d_R(\boldsymbol{c}, \boldsymbol{a'}) < d_R(\boldsymbol{a}, \boldsymbol{b}) + d_R(\boldsymbol{b}, \boldsymbol{c}) + d_R(\boldsymbol{c}, \boldsymbol{a})$. Also $\mathrm{Maj}(\boldsymbol{a'}, \boldsymbol{b}, \boldsymbol{c}) = \mathrm{Maj}(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}) \notin R$. But this contradicts our choice of $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}$.

Property 4 implies that the shortest paths to $\boldsymbol{b}$ and $\boldsymbol{c}$ diverge at $\boldsymbol{a}$, since for any shortest path to $\boldsymbol{b}$ the first variable flipped is from $U \cap V$ whereas for a shortest path to $\boldsymbol{c}$ it is from $W \cap V$. Similar statements hold for the vertices $\boldsymbol{b}$ and $\boldsymbol{c}$. Thus along the shortest path from $\boldsymbol{a}$ to $\boldsymbol{b}$ the first bit flipped is from $U \cap V$ and the last bit flipped is from $U \cap W$. On the other hand, if we go from $\boldsymbol{a}$ to $\boldsymbol{c}$ and then to $\boldsymbol{b}$, all the bits from $U \cap W$ are flipped before the bits from $U \cap V$. We use this crucially to define Q. We will add a set of 2-clauses that enforce the following rule on paths starting at $\boldsymbol{a}$: *Flip variables from $U \cap W$ before variables from $U \cap V$.* This will eliminate all shortest paths from $\boldsymbol{a}$ to $\boldsymbol{b}$ since they begin by flipping a variable in $U \cap V$ and end with $U \cap W$. The paths from $\boldsymbol{a}$ to $\boldsymbol{b}$ via $\boldsymbol{c}$ survive since they flip $U \cap W$ while going from $\boldsymbol{a}$ to $\boldsymbol{c}$ and $U \cap V$ while going from $\boldsymbol{c}$ to $\boldsymbol{b}$. However all remaining paths have length at least $|\boldsymbol{a} - \boldsymbol{b}| + 2$ since they flip twice some variables not in U.

Take all pairs of indices $\{(i, j) | i \in U \cap W, j \in U \cap V\}$. The following conditions hold from the definition of U, V, W: $a_i = \bar{c}_i = \bar{b}_i$ and $a_j = c_j = \bar{b}_j$. Add the 2-clause C_{ij} asserting that the pair of variables $x_i x_j$ must take values in $\{a_i a_j, c_i c_j, b_i b_j\} = \{a_i a_j, \bar{a}_i a_j, \bar{a}_i \bar{a}_j\}$. The new relation is $Q = R \wedge_{i,j} C_{ij}$. Note that $Q \subset R$. We verify that the distance between $\boldsymbol{a}$ and $\boldsymbol{b}$ in Q expands. It is easy to see that for any $j \in U$, the assignment $\boldsymbol{a} \oplus \boldsymbol{e}_j \notin Q$. Hence there are no shortest paths left from $\boldsymbol{a}$ to $\boldsymbol{b}$. On the other hand, it is easy to see that $\boldsymbol{a}$ and $\boldsymbol{b}$ are still connected, since the vertex $\boldsymbol{c}$ is still reachable from both.

Step 2. Isolating a pair of assignments whose distance expands.
The relation Q obtained in Step 1 may have several disconnected components. This

cleanup step isolates a single pair of assignments whose distance expands. By adding 2-clauses, we show that one can express a path of length $r+2$ between assignments at distance r.

Take $\boldsymbol{a}, \boldsymbol{b} \in Q$ whose distance expands in Q and $d_Q(\boldsymbol{a}, \boldsymbol{b})$ is minimized. Let $U = \{i | a_i \neq b_i\}$, and $|U| = r$. Shortest paths between $\boldsymbol{a}$ and $\boldsymbol{b}$ have certain useful properties:

1. *Each shortest path flips every variable from U exactly once.*
   Observe that each index $j \in U$ is flipped an odd number of times along any path from $\boldsymbol{a}$ to $\boldsymbol{b}$. Suppose it is flipped thrice along a shortest path. Starting at $\boldsymbol{a}$ and going along this path, let $\boldsymbol{b'}$ be the assignment reached after flipping j twice. Then the distance between $\boldsymbol{a}$ and $\boldsymbol{b'}$ expands, since j is flipped twice along a shortest path between them in Q. Also $d_Q(\boldsymbol{a}, \boldsymbol{b'}) < d_Q(\boldsymbol{a}, \boldsymbol{b})$, contradicting the choice of $\boldsymbol{a}$ and $\boldsymbol{b}$.
2. *Every shortest path flips exactly one variable $i \notin U$.*
   Since the distance between $\boldsymbol{a}$ and $\boldsymbol{b}$ expands, every shortest path must flip some variable $i \notin U$. Suppose it flips more than one such variable. Since $\boldsymbol{a}$ and $\boldsymbol{b}$ agree on these variables, each of them is flipped an even number of times. Let i be the first variable to be flipped twice. Let $\boldsymbol{b'}$ be the assignment reached after flipping i the second time. It is easy to verify that the distance between $\boldsymbol{a}$ and $\boldsymbol{b'}$ also expands, but $d_Q(\boldsymbol{a}, \boldsymbol{b'}) < d_Q(\boldsymbol{a}, \boldsymbol{b})$.
3. *The variable $i \notin U$ is the first and last variable to be flipped along the path.*
   Assume the first variable flipped is not i. Let $\boldsymbol{a'}$ be the assignment reached along the path before we flip i the first time. Then $d_Q(\boldsymbol{a'}, \boldsymbol{b}) < d_Q(\boldsymbol{a}, \boldsymbol{b})$. The distance between $\boldsymbol{a'}$ and $\boldsymbol{b}$ expands since the shortest path between them flips the variables i twice. This contradicts the choice of $\boldsymbol{a}$ and $\boldsymbol{b}$. Assume $j \in U$ is flipped twice. Then as before we get a pair $\boldsymbol{a'}, \boldsymbol{b'}$ that contradict the choice of $\boldsymbol{a}, \boldsymbol{b}$.

Every shortest path between $\boldsymbol{a}$ and $\boldsymbol{b}$ has the following structure: first a variable $i \notin U$ is flipped to $\bar{a}_i$, then the variables from U are flipped in some order, finally the variable i is flipped back to a_i.

Different shortest paths may vary in the choice of $i \notin U$ in the first step and in the order in which the variables from U are flipped. Fix one such path $T \subseteq Q$. Assume that $U = \{1, \ldots, r\}$ and the variables are flipped in this order, and the additional variable flipped twice is $r+1$. Denote the path by $\boldsymbol{a} \to \boldsymbol{u^0} \to \boldsymbol{u^1} \to \cdots \to \boldsymbol{u^r} \to \boldsymbol{b}$. Next we prove that we cannot flip the $r+1^{th}$ variable at an intermediate vertex along the path.

4. *For $1 \leq j \leq r-1$ the assignment $\boldsymbol{u^j} \oplus \boldsymbol{e_{r+1}} \notin Q$.* Suppose that for some j, we have $\boldsymbol{c} = \boldsymbol{u^j} \oplus \boldsymbol{e_{r+1}} \in Q$. Then $\boldsymbol{c}$ differs from $\boldsymbol{a}$ on $\{1, \ldots, i\}$ and from $\boldsymbol{b}$ on $\{i+1, \ldots, r\}$. The distance from $\boldsymbol{c}$ to at least one of $\boldsymbol{a}$ or $\boldsymbol{b}$ must expand, else we get a path from $\boldsymbol{a}$ to $\boldsymbol{b}$ through $\boldsymbol{c}$ of length $|\boldsymbol{a} - \boldsymbol{b}|$ which contradicts the fact that this distance expands. However $d_Q(\boldsymbol{a}, \boldsymbol{c})$ and $d_Q(\boldsymbol{b}, \boldsymbol{c})$ are strictly less than $d_Q(\boldsymbol{a}, \boldsymbol{b})$ so we get a contradiction to the choice of $\boldsymbol{a}, \boldsymbol{b}$.

We now construct the path of length $r+2$. For all $i \geq r+2$ we set $x_i = a_i$ to get a relation on $r+1$ variables. Note that $\boldsymbol{b} = \bar{a}_1 \ldots \bar{a}_r a_{r+1}$. Take $i < j \in U$. Along the path T the variable i is flipped before j so the variables $x_i x_j$ take one of three values

$\{a_i a_j, \bar{a}_i a_j, \bar{a}_i \bar{a}_j\}$. So we add a 2-clause C_{ij} that requires $x_i x_j$ to take one of these values and take $T = Q \wedge_{i,j} C_{ij}$. Clearly, every assignment along the path lies in T. We claim that these are the only solutions. To show this, take an arbitrary assignment $\boldsymbol{c}$ satisfying the added constraints. If for some $i < j \leq r$ we have $c_i = a_i$ but $c_j = \bar{a}_j$, this would violate C_{ij}. Hence the first r variables of $\boldsymbol{c}$ are of the form $\bar{a}_1 \dots \bar{a}_i a_{i+1} \dots a_r$ for $0 \leq i \leq r$. If $c_{r+1} = \bar{a}_{r+1}$ then $\boldsymbol{c} = \boldsymbol{u}^i$. If $c_{r+1} = a_{r+1}$ then $\boldsymbol{c} = \boldsymbol{u}^i \oplus \boldsymbol{e}_{r+1}$. By property 4 above, such a vector satisfies Q if and only if $i = 0$ or $i = r$, which correspond to $\boldsymbol{c} = \boldsymbol{a}$ and $\boldsymbol{c} = \boldsymbol{b}$ respectively.

Step 3. Structurally expressing paths of length 4.
Let $\mathcal{P}$ denote the set of all ternary relations whose graph is a path of length 4 between two assignments at Hamming distance 2. Up to permutations of coordinates, there are 6 such relations. Each of them is the conjunction of a 3-clause and a 2-clause. For instance, the relation $M = \{100, 110, 010, 011, 001\}$ can be written as $(x_1 \vee x_2 \vee x_3) \wedge (\bar{x}_1 \vee \bar{x}_3)$. (It is named so, because its graph looks like the letter 'M' on the cube.) These relations are "minimal" examples of relations that are not componentwise bijunctive. By projecting out intermediate variables from the path T obtained in Step 2, we structurally express one of the relations in $\mathcal{P}$. We structurally express other relations in $\mathcal{P}$ using this relation.

We will write all relations in $\mathcal{P}$ in terms of $M(x_1, x_2, x_3) = (x_1 \vee x_2 \vee x_3) \wedge (\bar{x}_1 \vee \bar{x}_3)$, by negating variables. For example $M(\bar{x}_1, x_2, x_3) = (\bar{x}_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \bar{x}_3) = \{000, 010, 110, 111, 101\}$.

Define the relation $P(x_1, x_{r+1}, x_2) = \exists x_3 \dots x_r \; T(x_1, \dots, x_{r+1})$. The table below listing all tuples in P and their witnesses, shows that the conditions for structural expressibility are satisfied, and $P \in \mathcal{P}$.

x_1, x_2, x_{r+1}	$x_3, \dots, x_r$
$a_1 a_2 a_{r+1}$	$a_3 \dots a_r$
$a_1 a_2 \bar{a}_{r+1}$	$a_3 \dots a_r$
$\bar{a}_1 a_2 \bar{a}_{r+1}$	$a_3 \dots a_r$
$\bar{a}_1 \bar{a}_2 \bar{a}_{r+1}$	$a_3 \dots a_k,\ \bar{a}_3 a_4 \dots a_r,\ \bar{a}_3 \bar{a}_4 a_5 \dots a_r\ \dots \bar{a}_3 \bar{a}_4 \dots \bar{a}_r$
$\bar{a}_1 \bar{a}_2 a_{r+1}$	$\bar{a}_3 \bar{a}_4 \dots \bar{a}_r$

Let $P(x_1, x_2, x_3) = M(l_1, l_2, l_3)$, where l_i is one of $\{x_i, \bar{x}_i\}$. We can now use P and 2-clauses to express every other relation in $\mathcal{P}$. Given $M(l_1, l_2, l_3)$ every relation in $\mathcal{P}$ can be obtained by negating some subset of the variables. Hence it suffices to show that we can express structurally $M(\bar{l}_1, l_2, l_3)$ and $M(l_1, \bar{l}_2, l_3)$ (M is symmetric in x_1 and x_3). In the following let λ denote one of the literals $\{y, \bar{y}\}$, such that it is $\bar{y}$ if and only if l_1 is $\bar{x}_1$.

$$
\begin{aligned}
M(\bar{l}_1, l_2, l_3) &= (\bar{l}_1 \vee l_2 \vee l_3) \wedge (l_1 \vee \bar{l}_3) \\
&= \exists y \; (\bar{l}_1 \vee \bar{\lambda}) \wedge (\lambda \vee l_2 \vee l_3) \wedge (l_1 \vee \bar{l}_3) \\
&= \exists y \; (\bar{l}_1 \vee \bar{\lambda}) \wedge (\lambda \vee l_2 \vee l_3) \wedge (l_1 \vee \bar{l}_3) \wedge (\bar{\lambda} \vee \bar{l}_3) \\
&= \exists y \; (\bar{l}_1 \vee \bar{\lambda}) \wedge (l_1 \vee \bar{l}_3) \wedge M(\lambda, l_2, l_3) \\
&= \exists y \; (\bar{l}_1 \vee \bar{\lambda}) \wedge (l_1 \vee \bar{l}_3) \wedge P(y, x_2, x_3)
\end{aligned}
$$

In the second step the clause $(\bar{\lambda} \vee \bar{l}_3)$ is implied by the resolution of the clauses $(\bar{l}_1 \vee \bar{\lambda}) \wedge (l_1 \vee \bar{l}_3)$.

For the next expression let λ denote one of the literals $\{y, \bar{y}\}$, such that it is negated if and only if l_2 is $\bar{x}_2$.

$$\begin{aligned}
M(l_1, \bar{l}_2, l_3) &= (l_1 \vee \bar{l}_2 \vee l_3) \wedge (\bar{l}_1 \vee \bar{l}_3) \\
&= \exists y\ (l_1 \vee l_3 \vee \lambda) \wedge (\bar{\lambda} \vee \bar{l}_2) \wedge (\bar{l}_1 \vee \bar{l}_3) \\
&= \exists y\ (\bar{\lambda} \vee \bar{l}_2) \wedge M(l_1, \lambda, l_3) \\
&= \exists y\ (\bar{\lambda} \vee \bar{l}_2) \wedge P(x_1, y, x_3)
\end{aligned}$$

The above expressions are both based on resolution and it is easy to check that they satisfy the properties of structural expressibility.

Step 4. Structurally expressing $\mathcal{S}_3$.
We structurally express $(x_1 \vee x_2 \vee x_3)$ from M using a formula derived from a gadget in [HD02]. This gadget expresses $(x_1 \vee x_2 \vee x_3)$ in terms of "Protected OR", which corresponds to our relation M.

$$\begin{aligned}
(x_1 \vee x_2 \vee x_3) &= \exists y_1 \dots y_5\ (x_1 \vee \bar{y}_1) \wedge (x_2 \vee \bar{y}_2) \wedge (x_3 \vee \bar{y}_3) \wedge (x_3 \vee \bar{y}_4) \\
&\quad \wedge M(y_1, y_5, y_3) \wedge M(y_2, \bar{y}_5, y_4) \qquad (2.4.1)
\end{aligned}$$

The table below listing the witnesses of each assignment for (x_1, x_2, x_3), shows that the conditions for structural expressibility are satisfied.

x_1, x_2, x_3	$y_1 \dots y_5$											
111	00011	00111	00110	00100	01100	01101	01001	11001	11000	10000	10010	10011
110							01001	11001	11000	10000		
100										10000		
101	00011	00111	00110	00100						10000	10010	10011
001	00011	00111	00110	00100								
011	00011	00111	00110	00100	01100	01101	01001					
010							01001					

From the relation $(x_1 \vee x_2 \vee x_3)$ we derive the other 3-clauses by reverse resolution, for instance

$$(\bar{x}_1 \vee x_2 \vee x_3) = \exists y\ (\bar{x}_1 \vee \bar{y}) \wedge (y \vee x_2 \vee x_3)$$

$\square$

Lemma 2.4.7 [GKMP09, Lemma 3.5] *Let $R \subseteq \{0,1\}^k$ be any relation of arity $k \geq 1$. R is structurally expressible from $\mathcal{S}_3$.*

The next theorem follows from the last two lemmas:

Theorem 2.4.8 (Structural expressibility theorem, modified from [GKMP09, Theorem 2.7]) *Let $\mathcal{S}$ be a finite set of logical relations. If $\mathcal{S}$ is not safely tight, then every logical relation is structurally expressible from $\mathcal{S}$.*

With Lemma 2.3.1, Corollary 2.4.5 and the preceding theorem, we can now complete the proofs for PSPACE-completeness and the exponential diameter:

Corollary 2.4.9 *If a finite set $\mathcal{S}$ of logical relations is not safely tight, then* st-Conn$_{\mathrm{C}}(\mathcal{S})$ *and* Conn$_{\mathrm{C}}(\mathcal{S})$ *are* PSPACE*-complete, and there exist* CNF$_{\mathrm{C}}(\mathcal{S})$*-formulas ϕ, such that the diameter of $G(\phi)$ is exponential in the number of variables.*

2.5 Safely Tight Sets of Relations: Structure and Algorithms

For safely tight sets of relations, the solution graphs possess certain structural properties that guarantee a linear diameter, and allow for P-algorithms for *st*-connectivity, and coNP-algorithms for connectivity. We start with safely componentwise bijunctive relations.

Lemma 2.5.1 [corrected from GKMP09, Lemma 4.3] *Let $\mathcal{S}$ be a set of safely componentwise bijunctive relations and φ a* $\mathrm{CNF_C}(\mathcal{S})$*-formula. If $\boldsymbol{a}$ and $\boldsymbol{b}$ are two solutions of φ that lie in the same component of $G(\varphi)$, then $d_\varphi(\boldsymbol{a}, \boldsymbol{b}) = |\boldsymbol{a} - \boldsymbol{b}|$, i.e., no distance expands.*

Proof. Consider first the special case in which every relation in $\mathcal{S}$ is bijunctive. In this case, φ is equivalent to a 2-CNF-formula and so the space of solutions of φ is closed under majority. We show that there is a path in $G(\varphi)$ from $\boldsymbol{a}$ to $\boldsymbol{b}$ such that along the path only the assignments on variables with indices from the set $D = \{i | a_i \neq b_i\}$ change. This implies that the shortest path is of length $|D|$ by induction on $|D|$. Consider any path $\boldsymbol{a} \to \boldsymbol{u}^1 \to \cdots \to \boldsymbol{u}^r \to \boldsymbol{b}$ in $G(\varphi)$. We construct another path by replacing u_i by $v_i = \mathrm{maj}(a, u_i, b)$ for $i = 1, \ldots, r$ and removing repetitions. This is a path because for any i v^i and v^{i+1} differ in at most one variable. Furthermore, v^i agrees with $\boldsymbol{a}$ and $\boldsymbol{b}$ for every i for which $a_i = b_i$. Therefore, along this path only variables in D are flipped.

For the general case, we show that every component F of $\mathrm{G}(\varphi)$ is the solution space of a 2-CNF-formula φ. Let $R \in \mathcal{S}$ be a safely componentwise bijunctive relation. Then any relation corresponding to a clause in φ (see Definition 2.1.2) of the form $R(x_1, \ldots, x_k)$ consists of bijunctive components $R_1, \ldots, R_m$. The projection of F onto $x_1, \ldots, x_k$ is itself connected and must satisfy R. Hence it lies within one of the components $R_1, \ldots, R_m$; assume it is R_1. We replace $R(x_1, \ldots, x_k)$ by $R_1(x_1, \ldots, x_k)$. Call this new formula φ_1. $G(\varphi_1)$ consists of all components of $\mathrm{G}(\varphi)$ whose projection on $x_1, \ldots, x_k$ lies in R_1. We repeat this for every clause. Finally we are left with a formula φ' over a set of bijunctive relations. Hence φ' is bijunctive and $G(\varphi')$ is a component of $G(\varphi)$. So the claim follows from the bijunctive case. $\square$

Corollary 2.5.2 [corrected from GKMP09, Corollary 4.4] *Let $\mathcal{S}$ be set of safely componentwise bijunctive relations. Then*

1. *for every $\phi \in \mathrm{CNF_C}(\mathcal{S})$ with n variables, the diameter of each component of $G(\phi)$ is bounded by n.*
2. $\textsc{st-Conn}_\mathrm{C}(\mathcal{S})$ *is in* P*.*
3. $\textsc{Conn}_\mathrm{C}(\mathcal{S})$ *is in* coNP*.*

The proof of this corollary in [GKMP09] is correct.

We now turn to safely OR-free relations; we need the following definition:

Definition 2.5.3 We define the *coordinate-wise partial order* $\leq$ on Boolean vectors as follows: $\boldsymbol{a} \leq \boldsymbol{b}$ if $a_i \leq b_i$, $\forall i$. A *monotone path* between two solutions $\boldsymbol{a}$ and $\boldsymbol{b}$ is a path $\boldsymbol{a} \to \boldsymbol{u}^1 \to \cdots \to \boldsymbol{u}^r \to \boldsymbol{b}$ in the solution graph such that $\boldsymbol{a} \leq \boldsymbol{u}^1 \leq \cdots \leq \boldsymbol{u}^r \leq \boldsymbol{b}$. A solution is *locally minimal* if it has no neighboring solution that is smaller than it.

Lemma 2.5.4 [corrected from GKMP09, Lemma 4.5] *Let $\mathcal{S}$ be a set of safely OR-free relations and φ a CNF$_C$($\mathcal{S}$)-formula. Every component of $G(\varphi)$ contains a minimum solution with respect to the coordinatewise order; moreover, every solution is connected to the minimum solution in the same component via a monotone path.*

Proof. We will show that there is exactly one such assignment in each component of $G(\varphi)$. Suppose there are two distinct locally minimal assignments $\boldsymbol{u}$ and $\boldsymbol{u'}$ in some component of $G(\varphi)$. Consider the path between them where the maximum Hamming weight of assignments on the path is minimized. If there are many such paths, pick one where the smallest number of assignments have the maximum Hamming weight. Denote this path by $\boldsymbol{u} = \boldsymbol{u^1} \to \boldsymbol{u^2} \to \cdots \to \boldsymbol{u^r} = \boldsymbol{u'}$. Let $\boldsymbol{u^i}$ be an assignment of largest Hamming weight in the path. Then $\boldsymbol{u^i} \neq \boldsymbol{u}$ and $\boldsymbol{u^i} \neq \boldsymbol{u'}$, since $\boldsymbol{u}$ and $\boldsymbol{u'}$ are locally minimal. The assignments $\boldsymbol{u^{i-1}}$ and $\boldsymbol{u^{i+1}}$ differ in exactly 2 variables, say, in x_1 and x_2. So $\{u_1^{i-1}u_2^{i-1},\ u_1^i u_2^i,\ u_1^{i+1}u_2^{i+1}\} = \{01, 11, 10\}$. Let $\hat{\boldsymbol{u}}$ be such that $\hat{u}_1 = \hat{u}_2 = 0$, and $\hat{u}_i = u_i$ for $i > 2$. If $\hat{\boldsymbol{u}}$ is a solution, then the path $\boldsymbol{u^1} \to \boldsymbol{u^2} \to \cdots \to \boldsymbol{u^i} \to \hat{\boldsymbol{u}} \to \boldsymbol{u^{i+1}} \to \cdots \to \boldsymbol{u^r}$ contradicts the way we chose the original path. Therefore, $\hat{\boldsymbol{u}}$ is not a solution. This means that there is a clause that is violated by it, but is satisfied by $\boldsymbol{u^{i-1}}$, $\boldsymbol{u^i}$, and $\boldsymbol{u^{i+1}}$. So the relation corresponding to that clause is not OR-free, thus $\mathcal{S}$ must have contained some not safely OR-free relation.

The unique locally minimal solution in a component is its minimum solution, because starting from any other assignment in the component, it is possible to keep moving to neighbors that are smaller, and the only time it becomes impossible to find such a neighbor is when the locally minimal solution is reached. Therefore, there is a monotone path from any satisfying assignment to the minimum in that component. □

Corollary 2.5.5 [corrected from GKMP09, Corollary 4.6] *Let $\mathcal{S}$ be a set of safely OR-free relations. Then*

1. *for every* CNF$_\mathrm{C}$($\mathcal{S}$)*-formula ϕ with n variables, the diameter of each component of $G(\phi)$ is bounded by $2n$.*
2. ST-CONN$_\mathrm{C}$($\mathcal{S}$) *is in* P.
3. CONN$_\mathrm{C}$($\mathcal{S}$) *is in* coNP.

The proof of this corollary in [GKMP09] is correct. Safely NAND-free relations are symmetric to safely OR-free relations, so that we have the following corollary which completes the proof of the dichotomy (Theorem 2.2.1).

Corollary 2.5.6 [corrected from GKMP09, Corollary 4.7] *Let S be a safely tight set of relations. Then*

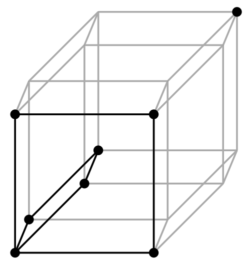
1. *for every $\phi \in$*CNF$_\mathrm{C}$($\mathcal{S}$) *with n variables, the diameter of each component of $G(\phi)$ is bounded by $2n$.*
2. ST-CONN$_\mathrm{C}$($\mathcal{S}$) *is in* P.
3. CONN$_\mathrm{C}$($\mathcal{S}$) *is in* coNP.

2.6 CPSS Sets of Relations: A Simple Algorithm for Connectivity

The rest of this chapter is devoted to the complexity of $\textsc{Conn}_\mathrm{C}(\mathcal{S})$. In this section we cover the tractable case; we show that for CPSS sets $\mathcal{S}$ of relations, for every $\mathrm{CNF}_\mathrm{C}(\mathcal{S})$-formula whose solution graph is disconnected, already the projection to the variables of some constraint is disconnected. We then use this property to derive a simple algorithm for $\textsc{Conn}_\mathrm{C}(\mathcal{S})$ (Gopalan et al. had given much more complicated algorithms in Lemmas 4.9, 4.10 and 4.13 of [GKMP09]).

Definition 2.6.1 A set $\mathcal{S}$ of logical relations is *constraint-projection separating* (*CPS*), if every $\mathrm{CNF}_\mathrm{C}(\mathcal{S})$-formula ϕ whose solution graph $G(\phi)$ is disconnected contains a constraint C_i s.t. $G(\phi_i)$ is disconnected, where ϕ_i is the projection of ϕ to $\mathrm{Var}(C_i)$.

For example, $\mathcal{S} = \{x \vee \overline{y}\}$ is CPS (for the proof see Lemma 2.6.4); so, e.g. for the $\mathrm{CNF}_\mathrm{C}(\mathcal{S})$-formula $(x \vee \overline{y}) \wedge (y \vee \overline{z}) \wedge (z \vee \overline{x})$, the projections to $\{x, y\}$, $\{y, z\}$ and $\{z, x\}$ are all disconnected.



In contrast, $\mathcal{S}' = \{x \vee \overline{y} \vee \overline{z}\}$ is not CPS: For example, the $\mathrm{CNF}_\mathrm{C}(\mathcal{S}')$-formula

$$(x \vee \overline{y} \vee \overline{z}) \wedge (y \vee \overline{z} \vee \overline{w}) \wedge (z \vee \overline{w} \vee \overline{x}) \wedge (w \vee \overline{x} \vee \overline{y})$$

is disconnected (see the graph on the right), but the projection to any three variables is connected.

We cannot provide an algorithm to decide for an arbitrary set of relations if it is CPS, but we will determine exactly which Schaefer sets are CPS, and also exhibit classes of non-Schaefer sets that are CPS.

For IHSB−, IHSB+ and bijunctive sets of relations we can prove even stronger properties in the next two lemmas:

Lemma 2.6.2 *Let $\mathcal{S}$ be a set of IHSB− (IHSB+) relations and ϕ a $\mathrm{CNF}_\mathrm{C}(\mathcal{S})$-formula. Then for any two components of $G(\phi)$, there is some constraint C_i of ϕ s.t. their images in the projection ϕ_i of ϕ to $\mathrm{Var}(C_i)$ are disconnected in $G(\phi_i)$.*

Proof. We prove the IHSB− case, the IHSB+ case is analogous. Consider any two components A and B of ϕ. Since every IHSB− relation is safely OR-free, there is a locally minimal solution $\boldsymbol{a}$ in A and a locally minimal solution $\boldsymbol{b}$ in B by Lemma 2.5.4. Let U and V be the sets of variables that are assigned 1 in $\boldsymbol{a}$ and $\boldsymbol{b}$, resp. At least one of the sets $U' = U \setminus V$ or $V' = V \setminus U$ is not empty, assume it is U'. Then for every $x_1 \in U'$ there must be a clause $x_1 \vee \overline{x}_2$ with $x_2 \in U$ since $\boldsymbol{a}$ is locally minimal, and also x_2 must be from U', else $\boldsymbol{b}$ would not be satisfying.

But then for x_2 there must be also some variable $x_3 \in U'$ and a clause $x_2 \vee \overline{x}_3$, and we can add the clause $x_1 \vee \overline{x}_3$ to ϕ without changing its value. Continuing this way, we will find a cycle, i.e. a clause $x_i \vee \overline{x}_{i+1}$ with $x_{i+1} = x_j$, $j < i$. But then we already have $x_j \vee \overline{x}_i$ added, thus $(s_i, s_j) \in \{(0,0), (1,1)\}$ for any solution $\boldsymbol{s}$ of ϕ, and there must be some constraint C_i with both x_i and x_j occurring in it (the C_i in which the original $x_i \vee \overline{x}_j$ appeared), and thus the projections of A and B to $\mathrm{Var}(C_i)$ are disconnected in

$G(\phi_i)$. □

Lemma 2.6.3 *Let $\mathcal{S}$ be a set of bijunctive relations and ϕ a $\mathrm{CNF_C}(\mathcal{S})$-formula. Then for any two components of $G(\phi)$, there is some constraint C_i of ϕ s.t. their images in the projection ϕ_i of ϕ to $\mathrm{Var}(C_i)$ are disconnected in $G(\phi_i)$.*

Proof. The proof is similar to the last one. Consider any two components A and B of ϕ and two solutions $\boldsymbol{a}$ in A and $\boldsymbol{b}$ in B that are at minimum Hamming distance. Let L be the set of literals that are assigned 1 in $\boldsymbol{a}$, but assigned 0 in $\boldsymbol{b}$. Then for every $l_1 \in L$ that is assigned 1 in $\boldsymbol{a}$, there must be a clause equivalent to $l_1 \vee \bar{l}_2$ in ϕ s.t. l_2 is also assigned 1 in $\boldsymbol{a}$, else the variable corresponding to l_1 could be flipped in $\boldsymbol{a}$, and the resulting vector would be closer to $\boldsymbol{b}$, contradicting our choice of $\boldsymbol{a}$ and $\boldsymbol{b}$. Also, l_2 must be assigned 0 in $\boldsymbol{b}$, i.e. $l_2 \in L$, else $\boldsymbol{b}$ would not be satisfying.

But then for l_2 there must be also some literal $l_3 \in L$ that is assigned 1 in $\boldsymbol{a}$ and a clause equivalent to $l_2 \vee \bar{l}_3$ in ϕ, and we can add the clause $l_1 \vee \bar{l}_3$ to ϕ without changing its value. Continuing this way, we will find a cycle, i.e. a clause equivalent to $l_n \vee \bar{l}_{n+1}$ with $l_{n+1} = l_m$, $m < n$. But then we already have $l_m \vee \bar{l}_n$ added, thus if x_i and x_j are the variables corresponding to l_n resp. l_m, then $(s_i, s_j) \in \{(0,1),(1,0)\}$ (if l_n and l_m were both positive or both negative), or $(s_i, s_j) \in \{(0,0),(1,1)\}$ (otherwise), for any solution $\boldsymbol{s}$ of ϕ. Also, there must be some constraint C_i with both x_i and x_j occurring in it (the constraint in which the clause equivalent to $l_n \vee \bar{l}_m$ appeared), and thus the projections of A and B to $\mathrm{Var}(C_i)$ are disconnected in $G(\phi_i)$. □

Lemma 2.6.4 *Every set $\mathcal{S}$ of safely componentwise bijunctive (safely componentwise IHSB−, safely componentwise IHSB+, affine) relations is constraint-projection separating.*

Proof. The affine case follows from the safely componentwise bijunctive case since every affine relation is safely componentwise bijunctive by Lemma 2.1.9.

If the relation corresponding to some C_i is disconnected, and there is more than one component of this relation for which ϕ has solutions with the variables of C_i assigned values in that component, the projection of ϕ to $\mathrm{Var}(C_i)$ must be disconnected in $G(\phi_i)$.

So assume that for every constraint C_i, ϕ only has solutions in which the variables of C_i are assigned values in one component P_i of the relation corresponding to C_i. Then we can replace every C_i with P_i to obtain an equivalent formula ϕ'. Since $\mathcal{S}$ is safely componentwise bijunctive (safely componentwise IHSB−, safely componentwise IHSB+), each P_i is bijunctive (IHSB−, IHSB−), and thus so is ϕ', and the statement follows from Lemmas Lemma 2.6.2 and Lemma 2.6.3. □

We are now ready to show how connectivity can be solved in polynomial time for CPSS sets of relations:

Lemma 2.6.5 *If a finite set $\mathcal{S}$ of relations is constraint-projection separating,* $\textsc{Conn}_\mathrm{C}(\mathcal{S})$ *is in* P^{NP}.[3] *If $\mathcal{S}$ is also Schaefer,* $\textsc{Conn}_\mathrm{C}(\mathcal{S})$ *is in* P.

Proof. For any $\mathrm{CNF_C}(\mathcal{S})$-formula ϕ, connectivity of $G(\phi)$ can be decided as follows:

[3] $\mathrm{P}^{\mathrm{NP}} = \mathrm{P}^{\mathrm{SAT}}$ is the class of languages decidable by a deterministic polynomial-time Turing machine with oracle-access to an NP-complete problem, e.g. Sat.

> For every constraint C_i of ϕ, obtain the projection ϕ_i of ϕ to the variables $\boldsymbol{x}^i$ occurring in C_i by checking for every assignment $\boldsymbol{a}$ of $\boldsymbol{x}^i$ whether $\phi[\boldsymbol{x}^i/\boldsymbol{a}]$ is satisfiable. Then $G(\phi)$ is connected iff for no ϕ_i, $G(\phi_i)$ is disconnected.

If $\mathcal{S}$ is Schaefer, every projection can be computed in polynomial time, else we use a SAT-oracle. Connectivity of every $G(\phi_i)$ can be checked in constant time.

If $G(\phi)$ is disconnected, some $G(\phi_i)$ is disconnected since ϕ is CPS by Lemma 2.6.4 below. If some $G(\phi_i)$ is disconnected, $G(\phi)$ clearly is also disconnected. □

Corollary 2.6.6 *If a finite set $\mathcal{S}$ of relations is CPSS,* $\text{Conn}_\text{C}(\mathcal{S})$ *is polynomial-time solvable.*

2.7 The Last Piece: coNP-Hardness for Connectivity

It remains to determine the complexity of Conn_C for safely tight sets of relations that are not CPSS. For non-Schaefer sets this was done already by Gopalan et al.:

Lemma 2.7.1 [corrected from GKMP09, Lemma 4.8] *For S safely tight, but not Schaefer,* $\text{Conn}_C(\mathcal{S})$ *is* coNP*-complete.*

Proof. The problem ANOTHER-SAT($\mathcal{S}$) is: given a formula φ in $\text{CNF}_\text{C}(\mathcal{S})$ and a solution $\boldsymbol{s}$, does there exist a solution $\boldsymbol{t} \neq \boldsymbol{s}$? Juban ([?], Theorem 2) shows that if $\mathcal{S}$ is not Schaefer, then ANOTHER-SAT is NP-complete. He also shows ([?], Corollary 1) that if $\mathcal{S}$ is not Schaefer, then the relation $x \neq y$ is expressible as a $\text{CNF}_\text{C}(\mathcal{S})$-formula.

Since $\mathcal{S}$ is not Schaefer, ANOTHER-SAT($\mathcal{S}$) is NP-complete. Let $\varphi, \boldsymbol{s}$ be an instance of ANOTHER-SAT on variables $x_1, \ldots, x_n$. We define a $\text{CNF}_\text{C}(\mathcal{S})$ formula ψ on the variables $x_1, \ldots, x_n, y_1, \ldots, y_n$ as

$$\psi(x_1, \ldots, x_n, y_1, \ldots, y_n) = \varphi(x_1, \ldots, x_n) \wedge_i (x_i \neq y_i)$$

It is easy to see that $G(\psi)$ is connected if and only if $\boldsymbol{s}$ is the unique solution to φ. □

We are now left with the case of Horn (dual Horn) sets of relations containing at least one relation that is not safely componentwise IHSB− (not safely componentwise IHSB+).

For one such set, namely $\{x \vee \overline{y} \vee \overline{z}\}$, Makino, Tamaki, and Yamamoto showed in 2007 that Conn_C is coNP-complete [MTY07]. Consequently, Gopalan et al. conjectured that Conn_C is coNP-complete for any such set, and already suggested a way for proving that: One had to show that $\text{Conn}_C(\{M\})$ for the relation $M = (x \vee \overline{y} \vee \overline{z}) \wedge (\overline{x} \vee z)$ is coNP-hard [GKMP09]. We will prove this in Lemma 2.7.9 by a reduction from the complement of a satisfiability problem.

Gopalan et al. stated (without giving the proof) that they could show that M is structurally expressible from every such set, using a similar reasoning as in the proof of their structural expressibility theorem (Lemma 3.4 in [GKMP09]). We give a quite different proof in Lemma 2.7.10, that shows that M actually is expressible as a $\text{CNF}_\text{C}(\mathcal{S})$-formula, which is of course a structural expression.

The proofs of Lemma 2.7.9 and Lemma 2.7.10 are arguably the most intricate part of this thesis and will be adapted for the no-constants and quantified cases in the next chapter.

2.7.1 Connectivity of Horn Formulas

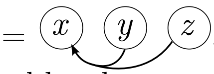
In this subsection, we introduce terminology and develop tools we will need for the proofs of Lemma 2.7.9 and Lemma 2.7.10.

Definition 2.7.2 Clauses with only one literal are called *unit clauses* (*positive* if the literal is positive, *negative* otherwise). Clauses with only negative literals are *restraint clauses*, and the sets of variables occurring in restraint clauses are *restraint sets*. Clauses having one positive and one or more negative literals are *implication clauses*. Implication clauses with two or more negative literals are *multi-implication clauses*.

A variable x *is implied by a set of variables* U, if setting all variables from U to 1 forces x to be 1 in any satisfying assignment. We write $\mathrm{Imp}(U)$ for the set of variables implied by U, we abbreviate $\mathrm{Imp}(\{x\})$ as $\mathrm{Imp}(x)$. We simply say that x *is implied,* if $x \in \mathrm{Imp}(U \setminus \{x\})$ for some U. Note that $U \subseteq \mathrm{Imp}(U)$ for all sets U.

U is *self-implicating* if every $x \in U$ is implied by $U \setminus \{x\}$. U is *maximal self-implicating*, if further $U = \mathrm{Imp}(U)$.

Remark 2.7.3. A Horn formula can be represented by a directed hypergraph with hyperedges of head-size one as follows: For every variable, there is a node, for every implication clause $y \vee \overline{x}_1 \vee \cdots \vee \overline{x}_k$, there is a directed hyperedge from $x_1, \ldots, x_k$ to y, for every restraint clause $\overline{x}_1 \vee \cdots \vee \overline{x}_k$, there is a directed hyperedge from $x_1, \ldots, x_k$ to a special node labeled "false", and for every positive unit clause x, there is a directed hyperedge from a special node labeled "true" to x.

We draw the directed hyperedges as joining lines, e.g., $x \vee \overline{y} \vee \overline{z} =$ 

.
For simplicity, we omit the "false" and "true" nodes in the drawings and let the corresponding hyperedges end, resp. begin, in the void.

For an introduction to directed hypergraphs see e.g. [GLPN93]. General CNF-formulas can be represented by general directed hypergraphs, see e.g. [AP01].

Lemma 2.7.4 *The solution graph of a Horn formula ϕ without positive unit clauses is disconnected iff ϕ has a locally minimal nonzero solution.*

Proof. This follows from Lemma 2.5.4 since the all-zero vector is a solution of every Horn formula without positive unit clauses, and Horn formulas are safely OR-free by Lemma 2.1.9. □

Lemma 2.7.5 *For every Horn formula ϕ without positive unit clauses, there is a bijection correlating each connected component ϕ_i with a maximal self-implicating set U_i containing no restraint set; U_i consists of the variables assigned 1 in the minimum solution of ϕ_i (the "lowest" component is correlated with the empty set).*

Proof. Let ϕ_i be a connected component of ϕ with minimum solution $\boldsymbol{s}$, and let U be the set of variables assigned 1 in $\boldsymbol{s}$. Since $\boldsymbol{s}$ is locally minimal, flipping any variable x_i from U to 0 results in a vector that is no solution, so there must be a clause in ϕ prohibiting that x_i is flipped. Since ϕ contains no positive unit-clauses, each $x_i \in U$ must appear as the positive literal in an implication clause with also all negated variables from U. It follows that U is self-implicating. Also, U must be maximal self-implicating and can contain no restraint set, else $\boldsymbol{s}$ were no solution.

Conversely, let U be a maximal self-implicating set containing no restraint set. Then the vector $\boldsymbol{s}$ with all variables from U assigned 1, and all others 0, is a locally minimal solution: All implication clauses $\overline{y}_1 \vee \cdots \vee \overline{y}_k \vee x$ with some $y_i \notin U$ are satisfied since $y_i = 0$, and for the ones with all $y_i \in U$, also $x \in U$ holds because U is maximal, so these are satisfied since $x = 1$. All restraint clauses are satisfied since U contains no restraint set. $\boldsymbol{s}$ is locally minimal since every variable assigned 1 is implied by U, so that any vector with one such variable flipped to 0 is no solution. By Lemma 4.5 of [GKMP09], every connected component has a unique locally minimal solution, so $\boldsymbol{s}$ is the minimum solution of some component. □

Corollary 2.7.6 *The solution graph of a Horn formula ϕ without positive unit clauses is disconnected iff ϕ has a non-empty maximal self-implicating set containing no restraint set.*

Lemma 2.7.7 *Let R_1 and R_2 be two connected components of a Horn relation R with minimum solutions $\boldsymbol{u}$ and $\boldsymbol{v}$, resp., and let U and V be the sets of variables assigned 1 in $\boldsymbol{u}$ and $\boldsymbol{v}$, resp. If then $U \subsetneq V$, no vector $\boldsymbol{a} \in R_1$ has all variables from V assigned 1.*

Proof. For the sake of contradiction, assume $\boldsymbol{a} \in R_1$ has all variables from V assigned 1. Then $\boldsymbol{a} \wedge \boldsymbol{v} = \boldsymbol{v}$, where $\wedge$ is applied coordinate-wise. Consider a path from $\boldsymbol{u}$ to $\boldsymbol{a}$, $\boldsymbol{u} \to \boldsymbol{w}^1 \to \cdots \to \boldsymbol{w}^k \to \boldsymbol{a}$. Since $U \subsetneq V$, we have $\boldsymbol{u} \wedge \boldsymbol{v} = \boldsymbol{u}$, so we can construct a path from $\boldsymbol{u}$ to $\boldsymbol{v}$ by replacing each $\boldsymbol{w}^i$ by $\boldsymbol{w}^i \wedge \boldsymbol{v}$ in the above path, and removing repetitions. Since R is Horn, it is closed under $\wedge$ (see Lemma 2.1.5), so all vectors of the constructed path are in R. But $\boldsymbol{u}$ and $\boldsymbol{v}$ are not connected in R, which is a contradiction. □

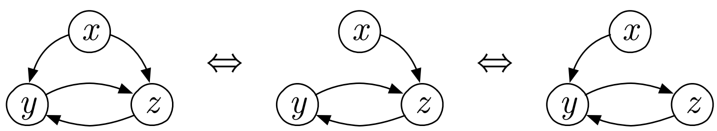
Definition 2.7.8 For a Horn formula ϕ, let $\nu(\phi)$ be the formula obtained from ϕ by recursively applying the following simplification rules as long as one is applicable; it is easy to check that the operations are equivalent transformations, and that the recursion must terminate:

(a) The constants 0 and 1 are eliminated in the obvious way.

(b) Multiple occurrences of some variable in a clause are eliminated in the obvious way.

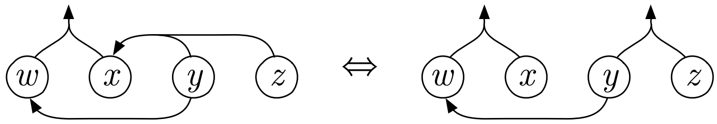
(c) If for some implication clause $c = x \vee \overline{y}_1 \vee \cdots \vee \overline{y}_k$ $(k \geq 1)$, x is already implied by $\{y_1, \ldots, y_k\}$ via other clauses, c is removed.

E.g., if there was a clause $x \vee \overline{z}_1 \vee \cdots \vee \overline{z}_l$ with $\{z_1, \ldots, z_l\} \subseteq \{y_1, \ldots, y_k\}$, or clauses $q \vee \overline{z}_1 \vee \cdots \vee \overline{z}_l$ and $x \vee \overline{q}$, c would be removed. Which clauses are removed by this rule may be random; e.g., for the formula $(\overline{x} \vee y) \wedge (\overline{x} \vee z) \wedge (\overline{z} \vee y) \wedge (\overline{y} \vee z)$, $\overline{x} \vee y$ *or* $\overline{x} \vee z$ would be removed:



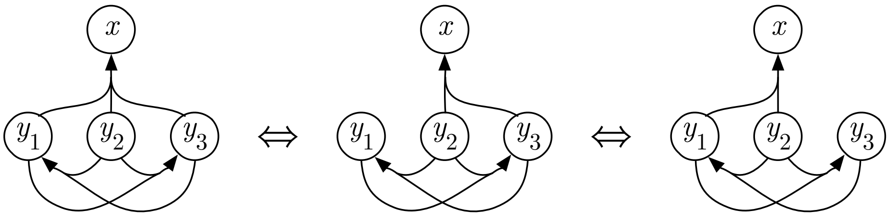
(d) If for some implication clause $c = x \vee \overline{y}_1 \vee \cdots \vee \overline{y}_k$ $(k \geq 1)$, $\mathrm{Imp}(\mathrm{Var}(c))$ contains a restraint set, c is replaced by $\overline{y}_1 \vee \cdots \vee \overline{y}_k$.

E.g., if there is a clause $\overline{r}_1 \vee \cdots \vee \overline{r}_l$ with $\{r_1, \ldots, r_l\} \subseteq \{x, y_1, \ldots, y_k\}$, or if there are clauses $q_1 \vee \overline{r}_1 \vee \cdots \vee \overline{r}_l$, $q_2 \vee \overline{r}_1 \vee \cdots \vee \overline{r}_l$ and $\overline{q}_1 \vee \overline{q}_2$. E.g., in the formula $(x \vee \overline{y} \vee \overline{z}) \wedge (w \vee \overline{y}) \wedge (\overline{w} \vee \overline{x})$, $x \vee \overline{y} \vee \overline{z}$ is replaced by $\overline{y} \vee \overline{z}$:



(e) If for some multi-implication clause $c = x \vee \overline{y}_1 \vee \cdots \vee \overline{y}_k$ $(k \geq 2)$, or for some restraint clause $d = \overline{y}_1 \vee \cdots \vee \overline{y}_k$, some $y_i \in \{y_1, \ldots, y_k\}$ is implied by $\{y_1, \ldots, y_k\} \setminus \{y_i\}$, the literal $\overline{y}_i$ is removed from c resp. d.

Which literals are removed by this rule may be random, as in the following example:



For a Horn relation R, let $\nu(R) = \nu(\phi)$ with some Horn formula ϕ representing R.

2.7.2 Reduction from Satisfiability

Lemma 2.7.9 $\textsc{Conn}_\mathrm{C}(\{M\})$ *with* $M = (x \vee \overline{y} \vee \overline{z}) \wedge (\overline{x} \vee z)$ *is* coNP*-hard.*

Proof. We reduce the no-constants satisfiability problem $\textsc{Sat}(\{P, N\})$ with $P = x \vee y \vee z$ and $N = \overline{x} \vee \overline{y}$ to the complement of $\textsc{Conn}_C(\{M\})$, where $M = (x \vee \overline{y} \vee \overline{z}) \wedge (\overline{x} \vee z)$. $\textsc{Sat}(\{P, N\})$ is NP-hard by Schaefer's dichotomy theorem (Theorem 2.4.1) since P is not 0-valid, not bijunctive, not Horn and not affine, while N is not 1-valid and not dual Horn.

Let ψ be any CNF$(\{P, N\})$-formula. If ψ only contains N-constraints, it is trivially satisfiable, so assume it contains at least one P-constraint. We construct a $\mathrm{CNF}_\mathrm{C}(\{M\})$-formula ϕ s.t. the solution graph $G(\phi)$ is disconnected iff ψ is satisfiable. First note that we can use the relations $\overline{x} \vee \overline{y} = M(0, x, y)$ and $\overline{x} \vee y = M(x, 0, y)$.

For every variable x_i of ψ $(i = 1, \ldots, n)$, there is the same variable x_i in ϕ. For every N-constraint $\overline{x}_i \vee \overline{x}_j$ of ψ, there is the clause $\overline{x}_i \vee \overline{x}_j$ in ϕ also. For every P-constraint $c_p = x_{i_p} \vee x_{j_p} \vee x_{k_p}$ $(p = 1, \ldots, m)$ of ψ there is an additional variable q_p in ϕ, and for every $x_l \in \{x_{i_p}, x_{j_p}, x_{k_p}\}$ appearing in c_p, there are two more additional variables a_{pl} and b_{pl} in ϕ. Now for every c_p, for each $l \in \{i_p, j_p, k_p\}$ the constraints $\overline{q}_p \vee a_{pl}$, $(\overline{x}_l \vee \overline{a}_{pl} \vee b_{pl}) \wedge \left(\overline{b}_{pl} \vee x_l\right)$ and $\overline{b}_{pl} \vee q_{(p+1) \bmod m}$ are added to ϕ. See the figures for examples of the construction.

If ψ is satisfiable, there is an assignment $\boldsymbol{s}$ to the variables x_i s.t. for every P-constraint c_p there is at least one $x_l \in \{x_{i_p}, x_{j_p}, x_{k_p}\}$ assigned 1, and for no N-constraint $\overline{x}_i \vee \overline{x}_j$, both x_i and x_j are assigned 1. We extend $\boldsymbol{s}$ to a locally minimal nonzero satisfying assignment $\boldsymbol{s}'$ for ϕ; then $G(\phi)$ is disconnected by Lemma 2.7.4: Let all $q_p = 1$, $a_{pl} = 1$, and all $b_{pl} = x_l$ in $\boldsymbol{s}'$. It is easy to check that all clauses of ϕ are satisfied, and that all variables assigned 1 appear as the positive literal in an implication clause with

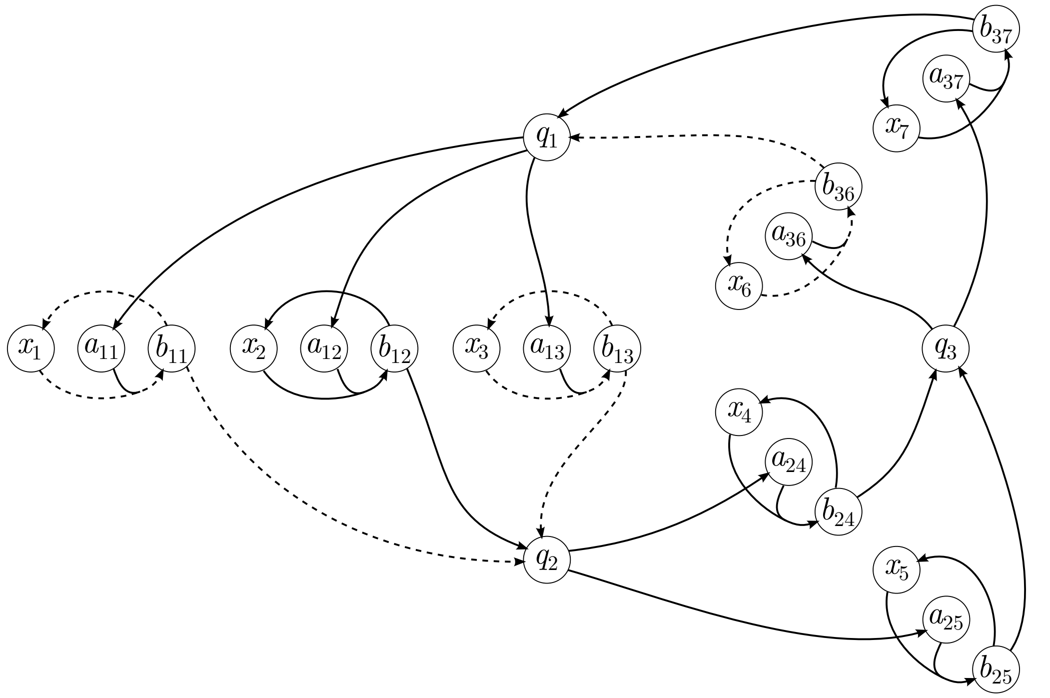
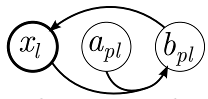


Figure 2.7.1 An example for the proof of Lemma 2.7.9, illustrating the idea. Depicted here is the hypergraph representation (see Remark 2.7.3) of ϕ for $\psi = (x_1 \vee x_2 \vee x_3) \wedge (x_4 \vee x_5) \wedge (x_6 \vee x_7)$, as constructed in the proof.

Any self-implicating set of ϕ must contain a "large circulatory", passing through each q_p and at least one gadget 

for each p; these gadgets act as "valves": If some x_i is not allowed to be assigned 1 (due to restraints), the circulatory may not pass through any gadget containing x_i.

Every maximal self-implicating set also contains all a_{pl}; here, for example, one maximal self-implicating set consist of the variables with the outgoing edges drawn solid.

If we would add restraint clauses to ψ s.t. ψ would become unsatisfiable, e.g. $\overline{x_4} \vee \overline{x_6}$, $\overline{x_4} \vee \overline{x_7}$, $\overline{x_5} \vee \overline{x_6}$, and $\overline{x_5} \vee \overline{x_7}$, each maximal self-implicating set of the corresponding ϕ would contain a restraint set, so that $G(\phi)$ would be connected.

all its variables assigned 1, so that $\boldsymbol{s}'$ is locally minimal. $\boldsymbol{s}'$ is nonzero since ψ contains at least one P-constraint.

Conversely, if $G(\phi)$ is disconnected, ϕ has a maximal self-implicating set U containing no restraint set by Corollary 2.7.6. It is easy to see that U must contain all q_p, all a_{pl}, and for every p for at least one $l \in \{i_p, j_p, k_p\}$ both b_{pl} and x_l(see also Figure 2.7.1 and the explanation beneath). Thus the assignment with all $x_i \in U$ assigned 1 and all other x_i assigned 0 satisfies ψ. □

2.7.3 Expressing *M*

Lemma 2.7.10 *The relation $M = (x \vee \overline{y} \vee \overline{z}) \wedge (\overline{x} \vee z)$ is expressible as a* $\mathrm{CNF_C}(\{R\})$*-formula for every Horn relation R that is not safely componentwise IHSB−.*

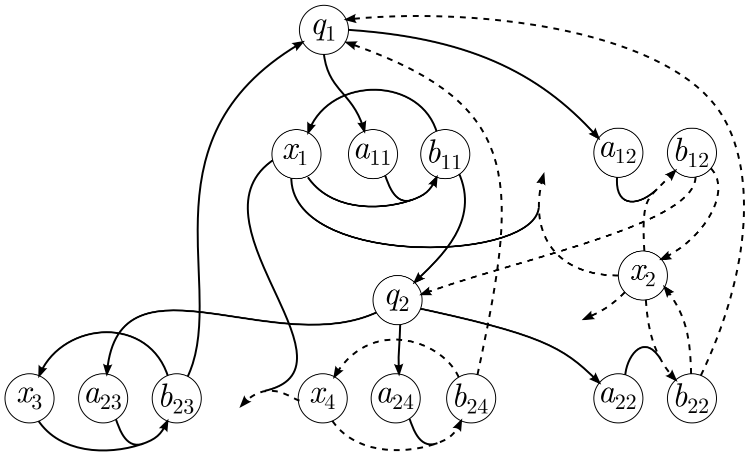
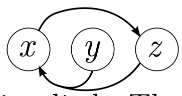
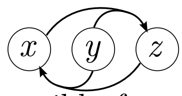


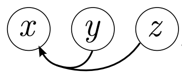
Figure 2.7.2 A more complex example, with a variable of ψ appearing twice in a P-constraint: Depicted is ϕ for $\psi = (x_1 \vee x_2) \wedge (x_3 \vee x_4 \vee x_2) \wedge (\overline{x}_1 \vee \overline{x}_2) \wedge (\overline{x}_1 \vee \overline{x}_4) \wedge (\overline{x}_2)$.

ψ is satisfiable with the unique solution $x_1 = x_3 = 1$ and $x_2 = x_4 = 0$, and $G(\phi)$ is disconnected (with exactly two components, since there is exactly one maximal self-implicating set containing no restraint set, consisting of the variables with the outgoing edges drawn solid).

Proof. $M = (x \vee \overline{y} \vee \overline{z}) \wedge (\overline{x} \vee z) =$ 

contains a multi-implication clause where some negated variable is not implied. The only other 3-ary such relations are (up to permutation of variables)

$L = (x \vee \overline{y} \vee \overline{z}) \wedge (\overline{x} \vee \overline{y} \vee z) =$ 

and $K = x \vee \overline{y} \vee \overline{z} =$ 

.

We show that $M, L,$ or K is expressible from R by substitution of constants and identification of variables. We can then express M from K or L as

$$M(x, y, z) \equiv K(x, y, z) \wedge K(z, x, x) \equiv L(x, y, z) \wedge L(z, x, x).$$

We will argue with formulas simplified according to Definition 2.7.8; let $\phi_0 = \nu(R)$. The following 7 numbered transformation steps generate K, L, or M from ϕ_0. After each transformation, we assume ν is applied again to the resulting formula; we denote the formula resulting from the i'th transformation step in this way by ϕ_i.

In the first three steps, we ensure that the formula contains a multi-implication clause where some variable is not implied, in the fourth step we trim the multi-implication clause to size 3, and in the last three steps we eliminate all remaining clauses and variables not occurring in K, L, or M. Our first goal is to produce a formula with a connected solution graph that is not IHSB$-$, which will turn out helpful.

1. *Obtain a not componentwise IHSB$-$ formula ϕ_1 from ϕ_0 by identification of variables.*

Let $[\phi_1^*]$ be a connected component of $[\phi_1]$ that is not IHSB$-$, and let U be the set of variables assigned 1 in the minimum solution of ϕ_1^*.

2. *Substitute 1 for all variables from U.*

The resulting formula ϕ_2 now contains no positive unit-clauses. Further, the component $[\phi_2^*]$ of $[\phi_2]$ resulting from $[\phi_1^*]$ is still not IHSB$-$, and it has the all-0 vector as minimum solution. We show that

$$\phi_2^* \equiv \nu\left(\phi_2 \wedge (\bigvee_{x\in V_1} \overline{x}) \wedge \cdots \wedge \left(\bigvee_{x\in V_k} \overline{x}\right)\right), \tag{2.7.1}$$

where $V_1, \ldots, V_k$ are the sets of variables assigned 1 in the minimum solutions $\boldsymbol{v}^1, \ldots, \boldsymbol{v}^k$ of the other components of $[\phi_2]$, and we specified the formula to be in normal form:

> For any solution $\boldsymbol{a}$ in the component with minimum solution $\boldsymbol{v}^i$ we have $\boldsymbol{a} \geq \boldsymbol{v}^i$ (see Definition 2.5.3), so all components other than $[\phi_2^*]$ are eliminated in the right-hand side of (2.7.1). By Lemma 2.7.7, no vector from $[\phi_2^*]$ is removed.

By Lemma 2.7.5, $V_1, \ldots, V_k$ are exactly the non-empty maximal self-implicating sets of ϕ_2 that contain no restraint set.

Clearly, ϕ_2^* is not IHSB$-$. However, we have no restraint clauses at our disposal to generate ϕ_2^* from ϕ_2; nevertheless, we can isolate a connected part of ϕ_2^* that is not IHSB$-$, as we will see.

Since ϕ_2^* is not IHSB$-$, it contains a multi-implication clause c^*, and by (2.7.1) it is clear that ϕ_2 must contain the same clause c^*.

By simplification rule (d), $\mathrm{Imp}(\mathrm{Var}(c^*))$ contains no restraint set in ϕ_2. Now if some self-implicating set U^* were implied by $\mathrm{Var}(c^*)$, the related maximal self-implicating set U_m^* (which then were also implied by $\mathrm{Var}(c^*)$) could contain no restraint set, thus a restraint clause would be added for the variables from U_m^* in (2.7.1). But then c^* would be removed by ν in (2.7.1), again due to rule (d), which is a contradiction. Thus $\mathrm{Imp}(\mathrm{Var}(c^*))$ also contains no self-implicating set in ϕ_2, and so the following operation eliminates all self-implicating sets and all restraint clauses:

3. *Substitute 0 for all remaining variables not implied by* $\mathrm{Var}(c^*)$.

This operation also produces no new restraint clauses since any implication clause with the positive literal not implied by $\mathrm{Var}(c^*)$ must also have some negative literal not implied by $\mathrm{Var}(c^*)$, and thus vanishes.

Further, since ϕ_2 contained no positive unit-clauses, the formula cannot have become unsatisfiable by this operation. Also, it is easy to see that the simplification initiated by the substitution of 0 for some variable x_i can only affect clauses c with $x_i \in \mathrm{Imp}(\mathrm{Var}(c))$, so c^* is retained in ϕ_3.

Since all variables not from $\mathrm{Var}(c^*)$ are now implied by $\mathrm{Var}(c^*)$, and $\mathrm{Imp}(\mathrm{Var}(c^*))$ is not self-implicating, c^* contains a variable that is not implied; w.l.o.g., let $c^* = x \vee \overline{y} \vee \overline{z}_1 \vee \cdots \vee \overline{z}_k$ $(k \geq 1)$ s.t. y is not implied.

4. *Identify* $z_1, \ldots, z_k$*, call the resulting variable* z*.*

This produces the clause $c^\sim = x \vee \overline{y} \vee \overline{z}$ from c^*. Clearly, y is still not implied in ϕ_4, and since x was not implied by any set $U \subsetneq \{y, z_1, \ldots, x_k\}$ by simplification rule (e) in ϕ_3, and no z_i was implied by y, it follows for ϕ_4 that

(*) $x \notin \mathrm{Imp}(y)$, $x \notin \mathrm{Imp}(z)$, $z \notin \mathrm{Imp}(y)$, y is not implied.

Also, since x was implied by $\{y, z_1, \dots, x_k\}$ only via c^* in ϕ_3 due to simplification rule (c), x is implied by $\{y, z\}$ only via $c^\sim$ in ϕ_4.

In the following steps, we eliminate all variables other than x, y, z, s.t. $c^\sim$ is retained and (*) is maintained. It follows that we are then left with K, L, or M, since the only clauses only involving x, y, z and satisfying (*) besides $c^\sim$ are from $\{z \vee \overline{x},\ z \vee \overline{x} \vee \overline{y}\}$.

5. *Substitute 1 for every variable from* $\mathrm{Imp}(y) \setminus \{y\}$.

For the simplification initiated by this operation, note that ϕ_4 contained no restraint clauses. It follows that the formula cannot have become unsatisfiable by this operation. Further, it is easy to see that for a Horn formula without restraint clauses, at a substitution of 1 for variables from a set U, only clauses c containing at least one variable $x_i \in \mathrm{Imp}(U)$ are affected by the simplification. Thus, $c^\sim$ is not affected since x, y and z were not implied by $\mathrm{Imp}(y) \setminus \{y\}$.

We must carefully check that (*) is maintained since substitutions of 1 may result in new implications: Since $\mathrm{Imp}(y) \setminus \{y\}$ is empty in ϕ_5, still $x \notin \mathrm{Imp}(y)$ and $z \notin \mathrm{Imp}(y)$. It is easy to see that x could only have become implied by z as result of transformation 5 if there had been a multi-implication clause (other than $c^\sim$) in ϕ_4 with the positive variable implying x, and each negated variable implied by y or z; but this is not the case since x was implied by $\{y, z\}$ only via $c^\sim$ in ϕ_4, thus still $x \notin \mathrm{Imp}(z)$.

We eliminate all remaining variables besides x, y, z by identifications in the next two steps. Since now $\mathrm{Imp}(y) \setminus \{y\}$ is empty, the only condition from (*) we have to care about is that $x \notin \mathrm{Imp}(z)$ remains true.

6. *Identify all remaining variables from* $\mathrm{Imp}(z) \setminus \{z\}$ *with* z.

Now $\mathrm{Imp}(z) \setminus \{z\}$ is empty, so the last step is easy:

7. *Identify all remaining variables other than* x, y, z *with* x.

□

This completes the coNP-completeness proof for connectivity and the proof of the trichotomy:

Corollary 2.7.11 *If a finite set* $\mathcal{S}$ *of logical relations is safely tight but not CPSS,* $\textsc{Conn}_{\mathrm{C}}(\mathcal{S})$ *is* coNP*-complete.*

Proof. By Lemma 2.5.6, $\textsc{Conn}_{\mathrm{C}}(\mathcal{S})$ is in coNP. If $\mathcal{S}$ is not Schaefer, coNP-hardness follows from Lemma 2.7.1. If $\mathcal{S}$ is Schaefer and not CPSS, it must be Horn and contain at least one relation that is not safely componentwise IHSB−, or dual Horn and contain at least one relation that is not safely componentwise IHSB+; in the first case, coNP-hardness follows from Lemmas 2.7.9 and 2.7.10, the second case follows by symmetry. □

2.8 Further Results about Constraint-Projection Separation

This section is not needed for the proof of the trichotomy (Theorem 2.2.2), but gives further insights that will be useful for the investigation of formulas without constants in the next chapter.

We start by showing that with Lemma 2.6.4, we have found all Schaefer sets of relations that are CPS:

Lemma 2.8.1 *If a set $\mathcal{S}$ of relations is Schaefer but not CPSS, it is not constraint-projection separating.*

Proof. Since $\mathcal{S}$ is Schaefer but not CPSS, it must contain some relation that is Horn but not safely componentwise IHSB−, or dual Horn but not safely componentwise IHSB+. Assume the first case, the second one is analogous. Then by Lemma 2.7.10, we can express $M = (x \vee \overline{y} \vee \overline{z}) \wedge (\overline{x} \vee z)$ as a $\mathrm{CNF_C}(\mathcal{S})$-formula. Consider the $\mathrm{CNF_C}(\mathcal{S})$-formula

$$T(u,v,w,x,y,z) = M(u,v,w) \wedge M(x,y,z) \wedge M(w,w,y) \wedge M(z,z,v)$$

$$\equiv ((u \vee \overline{v} \vee \overline{w}) \wedge (\overline{u} \vee w)) \wedge ((x \vee \overline{y} \vee \overline{z}) \wedge (\overline{x} \vee z)) \wedge (y \vee \overline{w}) \wedge (v \vee \overline{z})\,.$$

Now $G(T)$ is disconnected by Corollary 2.7.6 since $\{u,v,w,x,y,z\}$ is maximal self-implicating, but neither the projection $\exists x \exists y \exists z T \equiv M(u,v,w)$ to the variables of the first constraint in the $\mathrm{CNF}(\{M\})$-representation of T, nor the projection $\exists u \exists v \exists x \exists z T \equiv y \vee \overline{w}$ to the variables of the third one is disconnected. The second and fourth constraints are symmetric to the first and third ones.

Since in the $\mathrm{CNF_C}(\mathcal{S})$-representation of T every conjunct $M(r,s,t)$ of T $(r,s,t \in \{u,v,w,x,y,z\})$ is a $\mathrm{CNF_C}(\mathcal{S})$-formula $\bigwedge_i R_i(\boldsymbol{\xi}^i)$ with $R_i \in \mathcal{S}$ and $\xi^i_j \in \{0,1,r,s,t\}$, for every constraint C_i of T, the set $\mathrm{Var}(C_i)$ is a subset of $\{u,v,w\}$, $\{x,y,z\}$, $\{y,w\}$ or $\{v,z\}$, and thus also for no C_i the projection to $\mathrm{Var}(C_i)$ is disconnected. □

By Lemma 2.6.4 we see that there are non-Schaefer sets that are CPS, e.g. $\{R\}$ with $R = \{100, 010, 001\}$, which is safely componentwise bijunctive but not Schaefer. It is open whether there are other such sets not mentioned in Lemma 2.6.4.

While we will see in 3.1.2 that there are not safely tight sets that are no-constants CPS (see Definition 3.1.6), it is likely that no not safely tight set is CPS, else we had a $\mathrm{P^{NP}}$-algorithm for a PSPACE-complete problem. We can show that not safely tight sets are at least not by Lemma 2.6.4 CPS:

Lemma 2.8.2 *If a set of relations $\mathcal{S}$ is not safely tight, it also is not bijunctive, not safely componentwise IHSB−, not safely componentwise IHSB+, and not affine.*

Proof. By Lemma 2.1.9, $\mathcal{S}$ is not bijunctive and not affine. Also, $\mathcal{S}$ must contain a relation R s.t. the relation OR=$\{01, 10, 11\}$ can be obtained from R by identification of variables and substitution of constants. Since these operations are permutable, we can assume that OR can be obtained by first producing an $n-ary$ relation R' by identification of variables, and then w.l.o.g. setting the first $n-2$ variables to constants $c_1 \cdots c_{n-2}$. Then $\{c_1 \cdots c_{n-2}01, c_1 \cdots c_{n-2}10, c_1 \cdots c_{n-2}11\} \subset R'$, but $c_1 \cdots c_n 00 \notin R'$.

Since these 3 vectors from R' are in one component of R', already that component is not safely OR-free, so it cannot be Horn by Lemma 2.1.9, and thus is not IHSB−. But then R', and hence R, was not safely componentwise IHSB−. □

Remark 2.8.3. The Lemmas 2.6.2 and 2.6.3 cannot be generalized to safely componentwise bijunctive or safely componentwise IHSB− relations: For sets $\mathcal{S}$ of safely componentwise bijunctive (safely componentwise IHSB−) relations that are not bijunctive (IHSB−), there are $\mathrm{CNF_C}(\mathcal{S})$-formulas with pairs of components that are not disconnected in the projection to any constraint:

E.g., for the safely componentwise bijunctive relation $R = ((x \vee \overline{y}) \wedge \overline{z}) \vee (\overline{x} \wedge y \wedge z)$, the $\mathrm{CNF}(\{R\})$-formula $F(x, y, z, w) = R(x, y, z) \wedge R(y, x, w)$ has the four pairwise disconnected solutions a=0000, b=1100, c=0110, and d=1001, but a is connected to b in the projection to $\{x, y, z\}$ as well as in the one to $\{x, y, w\}$.

It follows that for such relations there is no algorithm for *st*-connectivity analogous to that of Lemma 2.6.5. In 3.1.2 we will actually see cases for no-constants formulas where we can solve connectivity in polynomial time via constraint-projection separation while *st*-connectivity is PSPACE-complete.

3 No-Constants and Quantified Variants

3.1 No-Constants

Complexity classifications for CNF($\mathcal{S}$)-formulas without constants seem to be more favored, but also more difficult to prove. For example, in [Sch78], Schaefer stated the no-constants classification as the main theorem, but then first proved a classification for formulas with constants as intermediate result. So we also will now attack the no-constants versions of our *st*-connectivity and connectivity problems, denoted by st-Conn($\mathcal{S}$) and Conn($\mathcal{S}$), respectively.

For *st*-connectivity and the diameter, we prove that the same dichotomy holds as for formulas with constants (in [GKMP06], Gopalan et al. already stated that they could extend their dichotomy theorem for *st*-connectivity to formulas without constants, but didn't show the proof).

For connectivity, we can extend the tractable class in two ways. Thereby, we get a quite interesting result: That there are cases when connectivity is easier than *st*-connectivity; namely, for some sets of relations, connectivity is in P even though *st*-connectivity is PSPACE-complete. While we can in some cases show that PSPACE-completeness and coNP-completeness carry over from the case with constants, we must leave open the complexity in two situations.

The following two theorems and the table below summarize our results.

<table>
<tr><th>$\mathcal{S}$</th><th>Conn($\mathcal{S}$)</th><th>Conn$_C$($\mathcal{S}$)</th><th>st-Conn($\mathcal{S}$)</th><th>st-Conn$_C$($\mathcal{S}$)</th></tr>
<tr><td>not safely tight, not *</td><td>PSPACE-c.</td><td rowspan="3">PSPACE-c.</td><td rowspan="3">PSPACE-c.</td><td rowspan="3">PSPACE-c.</td></tr>
<tr><td>not safely tight, not q.disc., *</td><td>in PSPACE</td></tr>
<tr><td>not safely tight, q.disc.</td><td>in P</td></tr>
<tr><td>safely tight, not *, not Schaefer</td><td rowspan="2">coNP-c.</td><td rowspan="6">coNP-c.</td><td rowspan="7">in P</td><td rowspan="7">in P</td></tr>
<tr><td>Schaefer, not implicative, not nc-CPSS</td></tr>
<tr><td>safely tight, *, not Schaefer, not q.disc., not nc-CPSS</td><td>in coNP</td></tr>
<tr><td>safely tight, q.disc, not CPSS</td><td rowspan="4">in P</td></tr>
<tr><td>nc-CPSS, not CPSS</td></tr>
<tr><td>implicative, not CPSS</td></tr>
<tr><td>CPSS</td><td>in P</td></tr>
</table>

Table 3.1 The classifications for CNF($\mathcal{S}$)-formulas without constants, in comparison to the case with constants.
* = (0-valid or 1-valid or complementive) q.disc. = quasi disconnecting
The cases where the complexity (up to polynomial-time isomorphisms) is not yet known are highlighted.

In this whole section, we assume the sets $\mathcal{S}$ to contain no empty relations (otherwise, since empty relations are not 0-valid and not 1-valid, the hardness statements for sets containing relations which are not 0-valid or not 1-valid would be wrong). This assumption has also to be made, e.g., for Schaefer's theorem in the no-constants case, but is often omitted.

Theorem 3.1.1 (Dichotomy theorem for st-Conn($\mathcal{S}$) and the diameter) *Let $\mathcal{S}$ be a finite set of logical relations.*

1. *If $\mathcal{S}$ is safely tight,* st-Conn($\mathcal{S}$) *is in* P*, and for every* CNF($\mathcal{S}$)*-formula ϕ, the diameter of $G(\phi)$ is linear in the number of variables.*
2. *Otherwise,* st-Conn($\mathcal{S}$) *is* PSPACE*-complete, and there are* CNF($\mathcal{S}$)*-formulas ϕ such that the diameter of $G(\phi)$ is exponential in the number of variables.*

Proof. See 3.1.1. □

Theorem 3.1.2 (Classification for Conn($\mathcal{S}$)) *Let $\mathcal{S}$ be a finite set of non-empty logical relations.*

1. *If $\mathcal{S}$ is nc-CPSS, quasi disconnecting or implicative,* Conn($\mathcal{S}$) *is in* P*.*
2. *Else, if $\mathcal{S}$ is Schaefer, or if $\mathcal{S}$ is safely tight but not Schaefer and not 0-valid nor 1-valid nor complementive,* Conn($\mathcal{S}$) *is* coNP*-complete.*
3. *Else, if $\mathcal{S}$ is safely tight,* Conn($\mathcal{S}$) *is in* coNP.
4. *Else, if $\mathcal{S}$ is not 0-valid nor 1-valid nor complementive,* Conn($\mathcal{S}$) *is* PSPACE*-complete.*
5. *Else,* Conn($\mathcal{S}$) *is in* PSPACE.

Proof. 1. See Corollary 3.1.10, Corollary 3.1.14, and Lemma 3.1.19.
2. See Lemma 3.1.20 and Corollary 3.1.28.
3. This result carries over from the case with constants (Theorem 2.2.2).
4. See Corollary 3.1.29.
5. This result carries over from the case with constants (Theorem 2.2.2). □

3.1.1 *st*-Connectivity and Diameter

The PSPACE-hardness proof for st-Conn($\mathcal{S}$) will be by reduction from st-Conn$_{\mathrm{C}}$($\mathcal{S}$). An obvious way to reduce a problem for formulas with constants to one for formulas without is to replace every occurrence of a constant with a new variable, and then to add constraints for the new variables such that, with regard to the problem at hand, the transformed formula is equivalent to the original one. This approach was already used by Schaefer [Sch78].

For *st*-connectivity, we have to make sure that for every two solutions of the original formula, there are two solutions of the transformed formula that are connected iff the solutions of the original formula are connected. In Lemma 3.1.5 below we show how this is possible for not safely tight sets of relations. We need the following definition and the next lemma.

Definition 3.1.3 A solution $\boldsymbol{a}$ of a formula ϕ is *isolated* if $\boldsymbol{a}$ is not connected to any other solution $\boldsymbol{b}$ in $G(\phi)$. A formula ϕ is *0-isolating* (*1-isolating*) if it has an isolated solution $\boldsymbol{a} \neq (1\cdots 1)$ ($\boldsymbol{a} \neq (0\cdots 0)$). Similarly, we define isolated vectors for relations, and 0-isolating and 1-isolating relations.

Lemma 3.1.4 *If an n-ary logical relation R is not safely OR-free, there is a 1-isolating* $\mathrm{CNF}(\{R\})$*-formula ϕ.*

Proof. By identification of variables, we can obtain a not OR-free relation R^* from R. W.l.o.g., assume that OR can be obtained from R^* by setting the last $n-2$ variables to constants $c_3, \ldots, c_n$ (for relations that also require identification of variables to obtain OR, the identification can be done in a prior step); then $R^*(x_1, x_2, c_3, \ldots, c_n) = x_1 \vee x_2$.

If $n = 2$, we take $\phi = R^*(x, x) = x$, then $[\phi] = \{1\}$.

Else, if all $c_3, \ldots, c_n = 1$, we define a 3-ary relation R' by identifying the last $n-2$ variables. Then $R'(x_1, x_2, 1) = x_1 \vee x_2$, and it follows that identifying the first two variables of R' yields a 2-ary relation $R'' = R'(x_1, x_1, x_2)$ with $11 \in R''$ and $01 \notin R''$, thus R'' equals $\{11, 00, 10\}$, $\{11, 00\}$, $\{11, 10\}$ or $\{11\}$. The second and fourth relation are already 1-isolating, so we let $\phi = R''(x_1, x_2)$. The first is $x_1 \vee \overline{x_2}$, and we obtain a 1-isolating relation by taking $\phi = R''(x_1, x_2) \wedge R''(x_2, x_1)$ with $[\phi] = \{11, 00\}$. From the third one we obtain $\{1\}$ as $\phi = R''(x_1, x_1)$.

Similarly, if all $c_3, \ldots, c_n = 0$, by identifying the last $n-2$ variables, and then the fist two, we get a relation R'' with $10 \in R''$ and $00 \notin R''$, thus R'' equals $\{10, 01, 11\}$, $\{10, 01\}$, $\{10, 11\}$ or $\{10\}$. Here again, the second and fourth relation are already 1-isolating, and from the first as well as from the third we obtain $\{1\}$ by identifying the two variables.

Otherwise, we define a 3-ary relation R'' by identifying all variables x_i of R^* with $c_i = 0$, then all with $c_i = 1$, and then the first two, i.e., $R''(x_1, x_2, x_3) = R(x_1, x_1, \xi_3, \ldots, \xi_n)$, where $\xi_i = x_2$ if $c_i = 1$ and $\xi_i = x_3$ if $c_i = 0$. Then $110 \in R''$ and $010 \notin R''$, and R'' is one of 64 possibles relations; Figure 3.1.1 shows how to produce a 1-isolating relation from each of them by identification of variables and conjunction. □

Lemma 3.1.5 *If a finite set of logical relations $\mathcal{S}$ is not safely tight, then* $\textsc{st-Conn}_{\mathrm{C}}(\mathcal{S})$ $\leq^p_m \textsc{st-Conn}(\mathcal{S})$*, and for every* $\mathrm{CNF}_{\mathrm{C}}(\mathcal{S})$*-formula ϕ with n variables and diameter d, there is a* $\mathrm{CNF}(\mathcal{S})$*-formula ϕ' with $O(n)$ variables and diameter $d' \geq d$.*

Proof. Since $\mathcal{S}$ is not safely tight, it must contain some relation that is not safely OR-free, so we can construct an n-ary 1-isolating relation R_1 by Lemma 3.1.4. Similarly, we can construct a m-ary 0-isolating relation R_0 from a not safely NAND-free relation. Let $\boldsymbol{a} \neq (1\cdots 1)$ be an isolated vector of R_0, and $\boldsymbol{b} \neq (0\cdots 0)$ an isolated vector of R_1; w.l.o.g. assume $a_1 = 0$ and $b_1 = 1$.

Now let $\phi(x_1, \ldots, x_n)$ be any $\mathrm{CNF}_{\mathrm{C}}(\mathcal{S})$-formula and $\boldsymbol{s}$ and $\boldsymbol{t}$ two solutions of ϕ. We construct a $\mathrm{CNF}(\mathcal{S})$-formula ϕ' by replacing every occurrence of the constant 0 in ϕ with a new variable y_1, and every occurrence of the constant 1 with a new variable z_1, and appending $\wedge R_0(y_1, y_2, \ldots, y_m) \wedge R_1(z_1, z_2, \ldots, z_n)$ to ϕ (where $y_2, \ldots, y_m$ and $z_2, \ldots, z_n$ are further new variables). Then $\boldsymbol{s} \cdot \boldsymbol{a} \cdot \boldsymbol{b}$ and $\boldsymbol{t} \cdot \boldsymbol{a} \cdot \boldsymbol{b}$ are connected in $G(\phi')$ iff $\boldsymbol{s}$ and $\boldsymbol{t}$ are connected in $G(\phi)$. This also shows that the maximal diameter carries over. □

```
{110}: already 1-isolating
{000 110}: already 1-isolating
{100 110}: identify x1,x2 -> {10}
{000 100 110}: R(x1,x2,x3) AND R(x2,x1,x3) = {000 110}
{110 001}: already 1-isolating
{000 110 001}: already 1-isolating
{100 110 001}: already 1-isolating
{000 100 110 001}: R(x1,x2,x3) AND R(x2,x1,x3) = {000 110 001}
{110 101}: already 1-isolating
{000 110 101}: already 1-isolating
{100 110 101}: identify x1,x2 -> {10}
{000 100 110 101}: R(x1,x2,x3) AND R(x2,x1,x3) = {000 110}
{110 001 101}: already 1-isolating
{000 110 001 101}: already 1-isolating
{100 110 001 101}: identify x1,x2 -> {10 01}
{000 100 110 001 101}: R(x1,x2,x3) AND R(x2,x1,x3) = {000 110 001}
{110 011}: already 1-isolating
{000 110 011}: already 1-isolating
{100 110 011}: already 1-isolating
{000 100 110 011}: already 1-isolating
{110 001 011}: already 1-isolating
{000 110 001 011}: already 1-isolating
{100 110 001 011}: identify x1,x2 -> {10 01}
{000 100 110 001 011}: R(x1,x2,x3) AND R(x1,x3,x2) = {000 100 011}
{110 101 011}: already 1-isolating
{000 110 101 011}: already 1-isolating
{100 110 101 011}: already 1-isolating
{000 100 110 101 011}: already 1-isolating
{110 001 101 011}: already 1-isolating
{000 110 001 101 011}: already 1-isolating
{100 110 001 101 011}: identify x1,x2 -> {10 01}
{000 100 110 001 101 011}: R(x1,x2,x3) AND R(x1,x3,x2) = {000 100 110 101 011}
{110 111}: identify x1,x2 -> {10 11}, then R(x1,x2) AND R(x2,x1) = {11}
{000 110 111}: identify x1,x2 -> {00 10 11}, then R(x1,x2) AND R(x2,x1) = {00 11}
{100 110 111}: identify x1,x2 -> {10 11}, then R(x1,x2) AND R(x2,x1) = {11}
{000 100 110 111}: identify x1,x2 -> {00 10 11}, then R(x1,x2) AND R(x2,x1) = {00 11}
{110 001 111}: already 1-isolating
{000 110 001 111}: identify x1,x3 -> {00 11}
{100 110 001 111}: already 1-isolating
{000 100 110 001 111}: identify x1,x3 -> {00 11}
{110 101 111}: identify x1,x2 -> {10 11}, then R(x1,x2) AND R(x2,x1) = {11}
{000 110 101 111}: identify x1,x2 -> {00 10 11}, then R(x1,x2) AND R(x2,x1) = {00 11}
{100 110 101 111}: identify x1,x2 -> {10 11}, then R(x1,x2) AND R(x2,x1) = {11}
{000 100 110 101 111}: identify x1,x2 -> {00 10 11}, then R(x1,x2) AND R(x2,x1) = {00 11}
{110 001 101 111}: identify x1,x3 -> {10 11}, then R(x1,x2) AND R(x2,x1) = {11}
{000 110 001 101 111}: identify x1,x3 -> {00 10 11}, then R(x1,x2) AND R(x2,x1) = {00 11}
{100 110 001 101 111}: identify x1,x3 -> {10 11}, then R(x1,x2) AND R(x2,x1) = {11}
{000 100 110 001 101 111}: identify x1,x3 -> {00 10 11}, then R(x1,x2) AND R(x2,x1) = {00 11}
{110 011 111}: identify x1,x2 -> {10 11}, then R(x1,x2) AND R(x2,x1) = {11}
{000 110 011 111}: identify x1,x2 -> {00 10 11}, then R(x1,x2) AND R(x2,x1) = {00 11}
{100 110 011 111}: identify x1,x2 -> {10 11}, then R(x1,x2) AND R(x2,x1) = {11}
{000 100 110 011 111}: identify x1,x2 -> {00 10 11}, then R(x1,x2) AND R(x2,x1) = {00 11}
{110 001 011 111}: identify x1,x3 -> {11}
{000 110 001 011 111}: identify x1,x3 -> {00 11}
{100 110 001 011 111}: identify x1,x3 -> {11}
{000 100 110 001 011 111}: identify x1,x3 -> {00 11}
{110 101 011 111}: identify x1,x2 -> {10 11}, then R(x1,x2) AND R(x2,x1) = {11}
{000 110 101 011 111}: identify x1,x2 -> {00 10 11}, then R(x1,x2) AND R(x2,x1) = {00 11}
{100 110 101 011 111}: identify x1,x2 -> {10 11}, then R(x1,x2) AND R(x2,x1) = {11}
{000 100 110 101 011 111}: identify x1,x2 -> {00 10 11}, then R(x1,x2) AND R(x2,x1) = {00 11}
{110 001 101 011 111}: identify x1,x3 -> {10 11}, then R(x1,x2) AND R(x2,x1) = {11}
{000 110 001 101 011 111}: identify x1,x3 -> {00 10 11}, then R(x1,x2) AND R(x2,x1) = {00 11}
{100 110 001 101 011 111}: identify x1,x3 -> {10 11}, then R(x1,x2) AND R(x2,x1) = {11}
{000 100 110 001 101 011 111}: identify x1,x3 -> {00 10 11}, then R(x1,x2) AND R(x2,x1) = {00 11}
```

Figure 3.1.1 Producing a 1-isolating relation from every 3-ary relation R satisfying $110 \in R$ and $010 \notin R$ for the last case of the proof of Lemma 3.1.4. This list is generated by the `main`-function of the class `Isolating` of SatConn.

We can now prove Theorem 3.1.1:

Proof of Theorem 3.1.1. 1. This result carries over from the case with constants (Theorem 2.2.1).

2. This follows from Theorem 2.2.1 with Lemma 3.1.5. □

3.1.2 Deciding Connectivity via Constraint-Projection Separation

We define the no-constants version of constraint-projection separation in the obvious way:

Definition 3.1.6 A set $\mathcal{S}$ of logical relations is *no-constants constraint-projection separating* (*nc-CPS*), if every CNF($\mathcal{S}$)-formula ϕ whose solution graph $G(\phi)$ is disconnected contains a constraint C_i s.t. $G(\phi_i)$ is disconnected, where ϕ_i is the projection of ϕ to $\mathrm{Var}(C_i)$.

Analogously to Lemma 2.6.5, and using Theorem 2.4.1, we immediately have the following result:

Lemma 3.1.7 *If a finite set $\mathcal{S}$ of relations is nc-CPS,* $\textsc{Conn}(\mathcal{S})$ *is in* P^{NP}. *If $\mathcal{S}$ also is 0-valid, 1-valid, or Schaefer,* $\textsc{Conn}(\mathcal{S})$ *is in* P.

Remark 3.1.8. The first part of the preceding lemma may be irrelevant, since connectivity is in coNP$\subseteq$ P^{NP} for all safely tight sets of relations by Lemma 2.5.6, and all not safely tight nc-CPS sets we know of are 0-valid and 1-valid (Lemma 3.1.13 below).

Since CPS sets of relations are also nc-CPS, using Lemma 2.6.4, we can extend the tractable class in the no-constants setting:

Definition 3.1.9 A set $\mathcal{S}$ of logical relations is *nc-CPSS*, if $\mathcal{S}$ is safely componentwise bijunctive, safely componentwise IHSB−, safely componentwise IHSB+, or affine, and if $\mathcal{S}$ also is 0-valid, 1-valid, or Schaefer.

Corollary 3.1.10 *If a finite set $\mathcal{S}$ of relations is nc-CPSS,* $\textsc{Conn}(\mathcal{S})$ *is in* P.

Example 3.1.11 $R = (x \vee y \vee z) \wedge (\overline{x} \vee \overline{y} \vee z) \wedge (\overline{x} \vee y \vee \overline{z}) = ((y \vee z) \wedge (\overline{x} \vee z) \wedge (\overline{x} \vee y)) \vee (x \wedge \overline{y} \wedge \overline{z})$ is not CPSS, but since R is 1-valid and e.g. safely componentwise bijunctive, $\textsc{Conn}(\{R\})$ is in P by the preceding corollary.

By Lemma 2.8.2 we see that all nc-CPSS sets of relations are also safely tight, but we can prove the following additional classes to be nc-CPS, containing sets that are not safely tight:

Definition 3.1.12 A set $\mathcal{S}$ of relations is *quasi componentwise bijunctive (quasi componentwise IHSB−, quasi componentwise IHSB+, quasi affine),* if the following holds for every relation R in $\mathcal{S}$:

- R is both 0-valid and 1-valid, and
- R is itself safely componentwise bijunctive (safely componentwise IHSB−, safely componentwise IHSB+, affine), or the following two conditions hold for R:

1. The all-0-vector is disconnected from the all-1-vector in $G(R)$.
2. the set $\mathcal{S}'$ of all relations producible from R by identification of variables (excluding R) is safely componentwise bijunctive (safely componentwise IHSB−, safely componentwise IHSB+, affine).

$\mathcal{S}$ is *quasi disconnecting*, if it is quasi componentwise bijunctive, quasi componentwise IHSB−, quasi componentwise IHSB+, or quasi affine.

Lemma 3.1.13 *If a finite set $\mathcal{S}$ of relations is quasi disconnecting, it is nc-CPS.*

Proof. Let ϕ be any CNF($\mathcal{S}$)-formula. First suppose that $\mathcal{S}$ contains a relation R where the all-0-vector is disconnected from the all-1-vector in $G(R)$, and that ϕ contains a constraint $R(x_1, \ldots, x_n)$ with all variables distinct. Then the projection ϕ_P of ϕ to $x_1, \ldots, x_n$ contains $0 \cdots 0$ and $1 \cdots 1$ as solutions since every constraint in ϕ is 0-valid and 1-valid. But $0 \cdots 0$ and $1 \cdots 1$ are disconnected in $G(R)$, thus $0 \cdots 0$ and $1 \cdots 1$ must also be disconnected in $G(\phi_P)$ (note that the solutions to ϕ_P are a subset of the solutions to $R(x_1, \ldots, x_n)$).

Otherwise, if ϕ contains no such constraint $R(x_1, \ldots, x_n)$, it is equivalent to a CNF($\mathcal{S}'$)-formula ϕ', where each constraint C_i of ϕ corresponds to an equivalent constraint C_i' of ϕ' with $\mathrm{Var}(C_i') = \mathrm{Var}(C_i)$. Thus since $\mathcal{S}'$ is CPS by Lemma 2.6.4, if ϕ is disconnected, there must be a constraint C_i of ϕ s.t. the projection of ϕ to $\mathrm{Var}(C_i)$ is disconnected. □

Corollary 3.1.14 *If a finite set $\mathcal{S}$ of relations is quasi disconnecting, there is a polynomial-time algorithm for* CONN($\mathcal{S}$).

Lemma 3.1.15 $R = \{0000, 0001, 0010, 1001, 1010, \ 1100, \ 0111, 1111\}$ *is not safely tight but quasi disconnecting.*

Proof. It is easy to check that R is not safely tight. But R is quasi componentwise bijunctive: Obviously, R is 0-valid, 1-valid and the all-0 vector is disconnected from the all-1 vector. It remains to show that the set $\mathcal{S}'$ of all relations producible from R by identification of variables is componentwise bijunctive. By identifying each pair of variables in turn we get all 3-ary relations in $\mathcal{S}'$:

- identifying variable 1 with 2 gives $\{000, 001, 010, 100, \ 111\}$.
- identifying variable 1 with 3 gives $\{000, 001, 100, \ 111\}$.
- identifying variable 1 with 4 gives $\{000, 001, 100, \ 111\}$.
- identifying variable 2 with 3 gives $\{000, 001, 011, 101, 111\}$.
- identifying variable 2 with 4 gives $\{000, 001, 011, 101, 111\}$.
- identifying variable 3 with 4 gives $\{000, \ 011, 110, 111\}$.

The connected components are signified; it is easy to check they are bijunctive by verifying that they are closed under maj (see Lemma 2.1.5). All 2-ary and 1-ary relations are automatically bijunctive. □

Remark 3.1.16. The `main`-function in the class `Sift` of SatConn enumerates all not safely tight relations that are quasi disconnecting. There are no 3-ary such relations, and up to permutation of variables and duality, the above relation is the only 4-ary one.

We now see that the complexity of connectivity and *st*-connectivity is “inverted” in some cases:

Corollary 3.1.17 *There are sets $\mathcal{S}$ of relations s.t.* Conn($\mathcal{S}$) *is in* P *while* st-Conn($\mathcal{S}$) *is* PSPACE-*complete.*

As in the case with constants, we have no algorithm to determine in general if a set of relations is nc-CPS, so one may discover yet more nc-CPS sets and thereby cases where connectivity is in P or in P^{NP}.

3.1.3 Deciding Connectivity via Self-Implication

While for formulas with constants, all sets of relations with a tractable connectivity problem are constraint-projection separating (assuming P≠coNP), without constants this is not true anymore: In Lemma 3.1.19 below we show that for Horn sets $\mathcal{S}$ that are 1-valid, there is a polynomial-time algorithm for Conn($\mathcal{S}$); now for example, $M = (x \vee \overline{y} \vee \overline{z}) \wedge (\overline{x} \vee z)$ is Horn and 1-valid but not CPSS, so by Lemma 2.8.1 it is not CPS, and from the proof of that lemma we find that M is also not nc-CPS.

Definition 3.1.18 A set $\mathcal{S}$ of relations is *implicative*, if it is Horn and 1-valid or dual Horn and 0-valid.

Horn relations that are 1-valid can contain no restraints, and without constants, restraints also cannot be produced from such relations. This makes it possible to decide connectivity in polynomial time:

Lemma 3.1.19 *If a finite set $\mathcal{S}$ of relations is implicative, there is a polynomial-time algorithm for* Conn($\mathcal{S}$).

Proof. We show the proof for $\mathcal{S}$ being Horn and 1-valid, the dual Horn and 0-valid case is symmetric. We can decide for any CNF($\mathcal{S}$)-formula ϕ whether $G(\phi)$ is connected as follows:.

- First assign all variables in positive unit-clauses; this produces a connectivity-equivalent formula ϕ'. Since $\mathcal{S}$ is 1-valid, ϕ' contains no restraints, so $G(\phi')$ is disconnected iff ϕ' has a non-empty self-implicating set by Corollary 2.7.6. The following polynomial-time algorithm finds the largest self-implicating set of ϕ' (which is the union of all self-implicating sets):

  (*) Let U be the set of all variables of ϕ'. Repeat the following as long as variables are removed:

  > For each $x \in U$, check if there is a clauses with x as the positive literal and all negated variables from U; if not, remove x from U.

- Now $G(\phi)$ is connected iff U is empty.

The correctness of algorithm (*) is easy to check by induction: At first, U includes every self-implicating set, and if U includes every self-implicating set, no variable from a self-implicating set is removed from U. Further, as long as U is not self-implicating, a variable *is* removed from U. □

It is tempting to extend this algorithm for formulas containing restraints, by checking for every maximal self-implicating set if it contains no restraint set. However, this seems to require checking an exponential number of possibilities to find all maximal self-implicating sets; this presumption is strongly supported by Lemma 3.1.20 below, which shows that connectivity is coNP-complete for such formulas.

3.1.4 coNP-Completeness for Connectivity within Schaefer

In this subsection we prove that CONN($\mathcal{S}$) is coNP-complete for all remaining Schaefer sets of relations[1]:

Lemma 3.1.20 *If $\mathcal{S}$ is a finite set of relations that is Schaefer but not nc-CPSS and not implicative,* CONN($\mathcal{S}$) *is* coNP*-complete.*

Proof. By Lemma 2.5.6, CONN($\mathcal{S}$) is in coNP. Since $\mathcal{S}$ is Schaefer but not nc-CPSS, it must be Horn and contain at least one relation that is not safely componentwise IHSB$-$, or dual Horn and contain at least one relation that is not safely componentwise IHSB$+$; since further $\mathcal{S}$ is not implicative, it must in the first case also contain at least one relation that is not 1-valid, and in the second case at least one that is not 0-valid. Thus in the first case, the statement follows from the Lemmas 3.1.22 and 3.1.23 below, the second case is symmetric. □

We need the following lemma:

Lemma 3.1.21 *If a non-empty logical relation R is Horn but not 1-valid, at least one of the relations* $\{0\}$ *or* $\{01\}$ *can be obtained from R by identification and permutation of variables.*

Proof. Since R is not 1-valid, but also not empty, there must be some vector $\boldsymbol{a} \in R$ with some $a_i = 0$. If $(0 \cdots 0) \in R$, we identify all variables and obtain $\{0\}$.

Otherwise, we define the relation R' by identifying all variables i with $a_i = 0$, and then all with $a_i = 1$. We show that R' equals $\{01\}$ or $\{10\}$:

Since $(0 \cdots 0)$ and $(1 \cdots 1)$ are not in R, $\{00\}$ and $\{11\}$ are not in R'. Further, since R is Horn, it is closed under $x \wedge y$ (see Lemma 2.1.5), and so the "to $\boldsymbol{a}$ complementary" vector $\boldsymbol{b} = \boldsymbol{a} \oplus \mathbf{1}$ is not in R, else $\boldsymbol{a} \wedge \boldsymbol{b} = (0 \cdots 0)$ were in R (where $\oplus$ and $\wedge$ are applied coordinate-wise). Thus if $\{01\} \in R'$, $\{10\} \notin R'$, and the other way around. □

The proof for coNP-hardness is by modifying the two corresponding Lemmas for formulas with constants (Lemma 2.7.10 and Lemma 2.7.9). While we cannot express the relation M from Lemma 2.7.10 as a CNF($\mathcal{S}$)-formula, we can assemble a CNF($\mathcal{S}$)-formula μ which is suitable for a reduction from satisfiability similar to the one of Lemma 2.7.9:

[1]Note that Schaefer sets of relations that are quasi disconnecting are also nc-CPSS or implicative.

Lemma 3.1.22 *If $\mathcal{S}$ is a finite set of non-empty Horn relations that contains at least one relation that is not 1-valid, and at least one relation that is not safely componentwise IHSB−, then there is a* $\mathrm{CNF}(\mathcal{S})$*-formula μ s.t. $\nu(\mu)$ contains no restraint clauses of size greater than 1 and no self-implicating sets, and s.t. $\nu(\mu)$ contains the unit-clause $\overline{v_0}$ and the clause $x \vee \overline{y} \vee \overline{z}$, s.t. y is not implied and x is implied only via the clause $x \vee \overline{y} \vee \overline{z}$.*

Proof. We modify the proof of Lemma 2.7.10.

Again, let $\phi_0 = \nu(R)$. The first transformation remains unchanged:

1. *Obtain a not componentwise IHSB− formula $\phi_1^{\sim}$ from ϕ_0 by identification of variables.*

We renamed the resulting formula since it will not directly be used as input for step 2; namely, we insert the following step:

#. Identify all variables in negative unit-clauses and call the resulting variable v_0, then identify all variables in positive unit-clauses and call the resulting variable v_1. If there were no negative unit-clauses, add the clauses $\overline{v_0}$ or $\overline{v_0} \wedge v_1$ obtained by Lemma 3.1.21 from a not 1-valid Horn relation.

The resulting formula $\phi_\#$ can be written as $\phi_\# = \phi_1 \wedge \overline{v_0}$ or $\phi_\# = \phi_1 \wedge \overline{v_0} \wedge v_1$, where ϕ_1 contains no unit-clauses, and s.t. v_0 and v_1 do not appear in ϕ_1. It is clear that $\phi_\#$ still is not componentwise IHSB−. The following transformations will only affect the part of the formula without unit-clauses, and our notation will be such that the (entire) formula resulting from step i is $\phi_i \wedge \overline{v_0}$ resp. $\phi_i \wedge \overline{v_0} \wedge v_1$.

We use ϕ_1 as input for the next transformation, which replaces step 2 of the original proof. Again, let $[\phi_1^*]$ be a connected component of $[\phi_1]$ that is not IHSB−, and let U be the set of variables assigned 1 in the minimum solution of ϕ_1^*.

2. *Identify all variables from U.*

We show that the resulting formula ϕ_2 has the same crucial properties as the formula ϕ_2 in proof of Lemma 2.7.10:

> W.l.o.g., assume U was not empty; then the vector $\boldsymbol{m}$ resulting from the minimum solution of ϕ_1^* has one variable x_m assigned 1 and all others 0. All vectors resulting from ϕ_1^* belong to the same component ϕ_2^*. In comparison to the relation ϕ_2^* of the original proof, ϕ_2^* contains additional vectors, resulting from vectors of ϕ_1 having all variables from U assigned 0.
> Since ϕ_1 contained no unit clauses, this also holds for ϕ_2, so ϕ_2 contains the all-0 vector, which is connected to $\boldsymbol{m}$. It follows that also here, ϕ_2^* has the all-0 vector as minimum solution. Further, ϕ_2^* also here is not componentwise IHSB−:
>
> > Let a, b, c be vectors from $[\phi_1^*]$ s.t. $a \wedge (b \vee c)$ is not in $[\phi_1^*]$. Since a, b, c all have 1 assigned to all variables from U, this also holds for $a \wedge (b \vee c)$, so for the vectors a', b', c' resulting from the identification in step 2, $a' \wedge (b' \vee c') \notin [\phi_2^*]$.

With this, the reasoning can proceed as in the original proof. We can perfectly simulate the next step (note that the entire formula after step 2 is $\phi_2 \wedge \overline{v_0}$ or $\phi_2 \wedge \overline{v_0} \wedge v_1$):

3. *Identify all variables of ϕ_2 not implied by Var(c^*) with v_0.*

It is clear that this has the same effect on ϕ_2 as a substitution with 0, so as in the original proof, we now have a clause $c = x \vee \overline{y} \vee \overline{z}_1 \vee \cdots \vee \overline{z}_k$ ($k \geq 3$) s.t. y is not implied. The next step remains unchanged:

4. *Identify $z_1, \dots, z_k$, call the resulting variable z.*

The resulting formula $\phi_4 \wedge \overline{v_0}$ resp. $\phi_4 \wedge \overline{v_0} \wedge v_1$ now contains the clause $x \vee \overline{y} \vee \overline{z}$, and y is not implied; further, it still contains no restraints of size greater than 1. However, step 4 may have produced self-implicating sets. Before we deal with this problem we append one more step to ensure that x is implied only via $x \vee \overline{y} \vee \overline{z}$.

Since $x \notin \mathrm{Imp}(y)$, $x \notin \mathrm{Imp}(z)$, and x is implied by $\{y, z\}$ only via $x \vee \overline{y} \vee \overline{z}$, each other implication clause with x as positive literal must have at least one negative literal with a variable not implied by $\{y, z\}$, so the following step eliminates all other clauses with x as positive literal:

5. *If a variable $x_i \notin \mathrm{Imp}(\{y, z\})$ appears in a negative literal of an implication clause having x as positive literal, identify x_i with x; repeat as long as there is such a variable.*

We take into account that also step 5 may have produced self-implicating sets (although that seems not possible).

To show how to eliminate the self-implicating sets which may have been produced in the last two steps, we first prove that ϕ_5 is still is not componentwise IHSB−:

> By the proof of Lemma 2.7.10, we know that we can express a not componentwise IHSB− relation (i.e., K, L, or M) from ϕ_5 by identification of variables and substitution of constants. We prove that any formula obtained from a componentwise IHSB− one by identification and substitution is also componentwise IHSB−; the statement then follows by reversal:
>
>> A relation obtained from a componentwise IHSB− one by identification of variables is componentwise IHSB− by definition. For substitution, consider a formula $\psi = \psi_1 \vee \cdots \vee \psi_k$ where $\psi_1, \dots, \psi_k$ are the connected components that can be written as IHSB− formulas. Then the formula ψ' resulting from the substitution is equivalent to $\psi'_1 \vee \cdots \vee \psi'_k$, where each ψ'_i is obtained from ϕ_i by substitution, and thus is IHSB− also (some ψ'_i may be empty). Since any two vectors of ψ' resulting from vectors of different components of ψ differ in at least two variables, all ψ'_i are disconnected. Now by Lemma 2.1.7, the connected components of each ϕ'_i are IHSB− since IHSB− relations are characterized by closure under an idempotent operation (see Lemma 2.1.5). It follows that ψ' is componentwise IHSB−.

Now if ϕ_5 contains self-implicating sets, we repeat steps 1 to 5 (skipping step #), with $\phi_5 \wedge \overline{v_0}$ resp. $\phi_5 \wedge \overline{v_0} \wedge v_1$ as input to step 1, until we obtain a formula containing no self-implicating sets. This leads to a formula with all the demanded properties since the input to step 1 always is not safely componentwise IHSB−, and since the recursion must terminate as variables are removed in every pass.

> *Remark:* We cannot execute or simulate the next step of Lemma 2.7.10 since the formula may contain no positive unit-clause, so we are not able to produce M. We could simplify the formula more, but this would be quite involved, and we can already use the formula for the reduction from satisfiability.

□

Lemma 3.1.23 $\textsc{Conn}(\{\mu\})$ *is* coNP*-hard for every Horn formula μ containing no restraint clauses of size greater than 1 and no self-implicating sets, and containing the unit-clause $\overline{v_0}$ and the clause $x \vee \overline{y} \vee \overline{z}$, s.t. y is not implied and x is implied only via the clause $x \vee \overline{y} \vee \overline{z}$.*

Proof. We modify the proof of Lemma 2.7.9: The $\mathrm{CNF_C}(\{M\})$-formula ϕ is replaced by a connectivity-equivalent $\mathrm{CNF}(\{\mu\})$-formula ϕ'; we assemble ϕ' by replacing the gadgets from which ϕ is built.

Clearly, μ can be written as $\mu = \lambda \wedge \varepsilon$, where ε contains only unit clauses (including $\overline{v_0}$), and λ contains no unit clauses and no variables appearing in unit clauses. Let the variables of λ be ordered such that x, y, z are the first three variables, in this order; let r be the arity of λ.

> *Remark:* The construction would be easy if we could structurally express M from μ; however, we see no easy way to that. ϕ' will satisfy a relation $\phi' = \exists \ldots \phi$, but this will be no structural expression in general.

We first show how to simulate the implication clause $\overline{u} \vee w$:

> Since λ contains no unit-clauses and no restraints, it has both the all-0- and the all-1-vector as solution.
> First assume that z is not implied by x. Starting from the all-0 vector, then setting all variables implied by x to 1, we see that $\boldsymbol{a} = (1, 0, 0, a_4, \ldots, a_r)$ is a solution to λ for some constants $a_4, \ldots, a_r$. By the clause $x \vee \overline{y} \vee \overline{z}$, the to $\boldsymbol{a}$ complementary vector $\boldsymbol{a} \oplus \mathbf{1}$ is no solution, so identifying all variables x_i where $a_i = 0$ with y, and all where $a_i = 1$ with x results in $x \vee \overline{y}$.
> Now assume that z is implied by x. Since x is not implied by z by simplification rule (c), there is some solution $\boldsymbol{a}$ with $a_1 = 0$ and $a_3 = 1$. Since z is implied by x, the to $\boldsymbol{a}$ complementary vector is no solution. So again, identifying all variables x_i with $a_i = 0$ with x, and all with $a_i = 1$ with z results in $\overline{x} \vee z$.
> It follows that we can in both cases express $(\overline{u} \vee w) \wedge \varepsilon$ as a $\mathrm{CNF}(\{\mu\})$-formula.

Now we construct ϕ', by replacing the gadgets from which the formula ϕ in the proof of Lemma 2.7.9 is built by $\mathrm{CNF}(\{\mu\})$-expressions; we make sure that all clauses of ϕ are entailed by the replacements, and no replacement contains a self-implicating set, which will turn out useful:

We replace $\overline{q}_p \vee a_{pl}$ by $\left(\overline{q}_p \vee a_{pl}\right) \wedge \varepsilon$, and $\overline{b}_{pl} \vee q_{(p+1) \bmod m}$ by $\left(\overline{b}_{pl} \vee q_{(p+1) \bmod m}\right) \wedge \varepsilon$.
We replace $\overline{x}_i \vee \overline{x}_j$ by $\overline{x_i} \vee v_0$ if $x_i = x_j$, and otherwise by

$$\lambda(v_0, x_i x_j, w_{ij4}, \ldots, w_{ijr}) \wedge \varepsilon, \tag{3.1.1}$$

where for each i, j, $w_{ij4}, \ldots, w_{ijr}$ are new variables. Clearly, $\overline{x}_i \vee \overline{x}_j$ is entailed by the replacement. Since no variables are identified in λ, there are no self-implicating sets in (3.1.1).

Finally, we replace $(\overline{x}_l \vee \overline{a}_{pl} \vee b_{pl}) \wedge \left(\overline{b}_{pl} \vee x_l\right)$ by

$$(\lambda(b_{pl}, a_{pl}, x_l, z_{pl4}, \ldots, z_{plr}) \wedge \varepsilon) \wedge \left(\left(\overline{b}_{pl} \vee x_l\right) \wedge \varepsilon\right), \tag{3.1.2}$$

where for each p, l, $z_{pl4}, \ldots, z_{plr}$ are new variables. By definition of λ, (3.1.2) contains the clause $\overline{x}_l \vee \overline{a}_{pl} \vee b_{pl}$. We show that (3.1.2) contains no self-implicating set:

> Clearly, we can ignore ε. Since no variables are identified in $\lambda(b_{pl}, a_{pl}, x_l, z_{pl4}, \ldots, z_{plr})$, this expression contains no self-implicating set. Since a_{pl} is not implied in (3.1.2), and b_{pl} is implied only via $\overline{x}_l \vee \overline{a}_{pl} \vee b_{pl}$, b_{pl} cannot belong to a self-implicating set in (3.1.2). Thus proceeding from $\lambda(b_{pl}, a_{pl}, x_l, z_{pl4}, \ldots, z_{plr})$ to the whole expression (3.1.2), it is easy to see that the clause $\overline{b}_{pl} \vee x_l$ cannot have produced a self-implicating set.

Now note that not only each clause $\overline{x}_i \vee \overline{x}_j$ of ϕ is entailed by its replacement, but

(*) the only restraint set of ϕ' is $\{v_0\}$, and a subset U of $\mathrm{Var}(\phi)$ implies v_0 in ϕ' exactly if U implies a restraint set $\{x_i, x_j\}$ in ϕ.

With this, we are ready to show that ϕ' is connected iff ϕ is connected, i.e., that ϕ' contains a maximal self-implicating set U containing no restraint set iff ϕ does:

Since ϕ' contains all implication clauses of ϕ, ϕ' also contains all self-implicating sets of ϕ. By (*), each maximal self-implicating set containing no restraint set of ϕ can be extended to a maximal self-implicating set containing no restraint set of ϕ'.

For the converse, first recall that all used gadgets are guaranteed to contain no self-implicating sets, and note that all additional variables of ϕ' except v_0 (which implies no other variable) each appear in only one gadget, so that no "shortcuts" are introduced. With this, a consideration analogously to the one for ϕ in the original proof shows that any maximal self-implicating set U' of ϕ' must contain all q_p, all a_{pl}, and for every p for at least one $l \in \{i_p, j_p, k_p\}$ both b_{pl} and x_l, so that U' must contain some maximal self-implicating set U of ϕ as subset. By (*), if U' contains no restraint set, the same holds for U. □

3.1.5 Reductions for Connectivity

The $\mathrm{CNF}(\mathcal{S})$-formula ϕ' constructed from a $\mathrm{CNF_C}(\mathcal{S})$-formula ϕ in the proof of Lemma 3.1.5 using 0- and 1-isolating relations may contain multiple components even if ϕ has only one component, so that construction cannot be used for the connectivity problem; But if we use relations with unique solutions instead, the number of components is retained, so that analogous to Lemma 3.1.5, a reduction is possible:

Definition 3.1.24 A formula ϕ is *0-unique* (*1-unique*) if it has exactly one solution $\boldsymbol{a}$ s.t. $\boldsymbol{a} \neq (1 \cdots 1)$ ($\boldsymbol{a} \neq (0 \cdots 0)$).

Lemma 3.1.25 *Let $\mathcal{S}$ be a finite set of logical relations. If there is a 0-unique and a 1-unique* $\mathrm{CNF}(\mathcal{S})$*-formula, then* $\textsc{Conn}_\mathrm{C}(\mathcal{S}) \leq_m^p \textsc{Conn}(\mathcal{S})$.

Using a result from Creignou et al., we can determine exactly for which $\mathcal{S}$ this is the case:

Lemma 3.1.26 *Let $\mathcal{S}$ be a finite set of logical relations. There is a 0-unique and a 1-unique* CNF$(\mathcal{S})$*-formula exactly if $\mathcal{S}$ contains at least one relation that is not 0-valid, at least one relation that is not 1-valid, and at least one relation that is not complementive.*

Proof. It is easy to see that if every relation in $\mathcal{S}$ is 0-valid, there is no 1-unique CNF$(\mathcal{S})$-formula, if every relation is 1-valid, there is no 0-unique one, and if if every relation in $\mathcal{S}$ is complementive, there is neither a 0-unique nor a 1-unique CNF$(\mathcal{S})$-formula.

Otherwise, Lemma 4.13 of [CH96] shows that in this case, $\overline{x} \wedge y$ is expressible as a CNF$(\mathcal{S})$-formula, which is both 0-unique and 1-unique. □

Corollary 3.1.27 *Let $\mathcal{S}$ be a finite set of logical relations. If $\mathcal{S}$ contains at least one relation that is not 0-valid, at least one relation that is not 1-valid, and at least one relation that is not complementive, then* $\textsc{Conn}_{\mathrm{C}}(\mathcal{S}) \leq_m^p \textsc{Conn}(\mathcal{S})$.

So for such sets $\mathcal{S}$, we can transfer the hardness results for formulas with constants (Theorem 2.2.2) to the no-constants case:

Corollary 3.1.28 *If $\mathcal{S}$ is a finite set of relations that is safely tight but not Schaefer and contains at least one relation that is not 0-valid, at least one relation that is not 1-valid, and at least one relation that is not complementive,* $\textsc{Conn}(\mathcal{S})$ *is* coNP*-complete.*

Corollary 3.1.29 *If $\mathcal{S}$ is a finite set of relations that is not safely tight and contains at least one relation that is not 0-valid, at least one relation that is not 1-valid, and at least one relation that is not complementive,* $\textsc{Conn}(\mathcal{S})$ *is* PSPACE*-complete.*

Here we end our investigation of no-constants formulas. It is easy to check that thus, the complexity of $\textsc{Conn}(\mathcal{S})$ remains open for sets $\mathcal{S}$ that are 0-valid, 1-valid, or complementive, but not Schaefer, nor nc-CPSS, nor quasi disconnecting (See also Table 3.1).

Example 3.1.30 The relation $R_{\mathrm{NAE}} = \{0,1\}^3 \setminus \{000, 111\}$ is complementive, but not safely tight and not quasi disconnecting, so we only know that $\textsc{Conn}(\mathcal{S})$ is in PSPACE.

3.2 Quantified Constraints

Now we look in the other direction and examine connectivity for more powerful versions of $\mathrm{CNF}_{\mathrm{C}}(\mathcal{S})$-formulas by allowing quantifiers. Since it is easy to transform any quantified formula into one in prenex normal form, we will assume all formulas to have that form, i.e.

$$Q_1 y_1 \cdots Q_m y_m \phi(y_1, \ldots, y_m, x_1, \ldots, x_n),$$

where ϕ is a $\mathrm{CNF}_{\mathrm{C}}(\mathcal{S})$-formula, and $Q_1, \ldots, Q_m \in \{\exists, \forall\}$ are quantifiers. We call these expressions *Q-CNF$_C$($\mathcal{S}$)-formulas* and denote the corresponding connectivity resp. *st*-connectivity problems by Q-$\textsc{Conn}_{\mathrm{C}}(\mathcal{S})$ resp. $\textsc{st}$-Q-$\textsc{Conn}_{\mathrm{C}}(\mathcal{S})$; the solution graph only involves the free variables$x_1, \ldots, x_n$.

Here, we can present a complete classification for both connectivity problems and the diameter, stated in the following two theorems and summarized in the table below:

<table>
<tr><th>$\mathcal{S}$</th><th>Q-$\textsc{Conn}_{\mathrm{C}}(\mathcal{S})$</th><th>$\textsc{Conn}_{\mathrm{C}}(\mathcal{S})$</th><th>$\textsc{st}$-Q-$\textsc{Conn}_{\mathrm{C}}(\mathcal{S})$</th><th>$\textsc{st}$-$\textsc{Conn}_{\mathrm{C}}(\mathcal{S})$</th></tr>
<tr><td>not safely tight</td><td rowspan="2">PSPACE-c.</td><td>PSPACE-c.</td><td rowspan="2">PSPACE-c.</td><td>PSPACE-c.</td></tr>
<tr><td>safely tight, not Schaefer</td><td rowspan="2">coNP-c.</td><td rowspan="4">in P</td></tr>
<tr><td>Horn, not c. I− /
dual Horn, not c. I+</td><td rowspan="2">coNP-c.</td><td rowspan="3">in P</td></tr>
<tr><td>Horn, c. I−, not I− /
dual Horn, c. I+, not I+</td><td rowspan="2">in P</td></tr>
<tr><td>bijunctive / affine / I− / I+</td><td>in P</td></tr>
</table>

Table 3.2 The classifications for Q-$\mathrm{CNF}_{\mathrm{C}}(\mathcal{S})$-formulas, in comparison to the case without quantifiers.
c. = componentwise I− = IHSB− I+ = IHSB+

Theorem 3.2.1 (Dichotomy theorem for Q-$\mathrm{CNF}_{\mathrm{C}}(\mathcal{S})$-formulas) *Let $\mathcal{S}$ be a finite set of logical relations.*

1. *If $\mathcal{S}$ is Schaefer,* $\textsc{st}$-Q-$\textsc{Conn}_{\mathrm{C}}(\mathcal{S})$ *is in* P, Q-$\textsc{Conn}_{\mathrm{C}}(\mathcal{S})$ *is in* coNP, *and for every Q-CNF$_C$($\mathcal{S}$)-formula ϕ, the diameter of $G(\phi)$ is linear in the number of free variables.*
2. *Otherwise, both* $\textsc{st}$-Q-$\textsc{Conn}_{\mathrm{C}}(\mathcal{S})$ *and* Q-$\textsc{Conn}_{\mathrm{C}}(\mathcal{S})$ *are* PSPACE*-complete, and there are Q-CNF$_C$($\mathcal{S}$)-formulas ϕ, such that the diameter of $G(\phi)$ is exponential in the number of free variables.*

Proof. See 3.2.1. □

Theorem 3.2.2 (Trichotomy theorem for Q-$\textsc{Conn}_{\mathrm{C}}(\mathcal{S})$) *Let $\mathcal{S}$ be a finite set of logical relations.*

1. *If $\mathcal{S}$ is bijunctive, IHSB−, IHSB+ or affine,* Q-$\textsc{Conn}_{\mathrm{C}}(\mathcal{S})$ *is in* P.
2. *Else if $\mathcal{S}$ is Schaefer,* Q-$\textsc{Conn}_{\mathrm{C}}(\mathcal{S})$ *is* coNP*-complete.*
3. *Else,* Q-$\textsc{Conn}_{\mathrm{C}}(\mathcal{S})$ *is* PSPACE*-complete.*

Proof. 1. See Lemmas 3.2.7, 3.2.8, and 3.2.9.
2. See Corollary 3.2.6.
3. This follows from Theorem 3.2.1. □

3.2.1 Properties that Persist

Recall from Lemma 2.1.5 that bijunctive, Horn / dual Horn, affine, and IHSB− / IHSB+ relations are characterized by closure properties. We begin by showing that these properties are retained when we quantify over some variables.

Lemma 3.2.3 *Let R be a logical relation that is closed under the coordinate-wise application of some operation f.*

1. *The relation obtained by quantifying existentially over some variable of R is also closed under f.*
2. *If f is not constant, the relation obtained by quantifying universally over some variable of R is also closed under f.*

Proof. 1. Let R be a $n+1$-ary relation, consisting of m vectors $(a^i, b^i_1, \ldots, b^i_n)$, $i = 1, \ldots, m$, that is closed under the coordinate-wise application of the k-ary relation f, i.e.

$$\left(f(a^{i_1}, \ldots, a^{i_k}),\ f(b^{i_1}_1, \ldots, b^{i_k}_1), \ldots,\ f(b^{i_1}_n, \ldots, b^{i_k}_n)\right) \in R$$

for all $1 \leq i_1, \ldots, i_k \leq m$. Let the relation $R' = \exists x\, R(x, \boldsymbol{y})$ be obtained, w.l.o.g., by quantifying existentially over the first variable. If then $\boldsymbol{b}^{i_1}, \ldots, \boldsymbol{b}^{i_k} \in R'$, also $\left(f(b^{i_1}_1, \ldots, b^{i_k}_1), \ldots,\ f(b^{i_1}_n, \ldots, b^{i_k}_n)\right) \in R'$ since for each $\boldsymbol{b}^i \in R'$, $(0, \boldsymbol{b}^i) \in R$ or $(1, \boldsymbol{b}^i) \in R$. Thus R also contains

$$\left(0,\ f(b^{i_1}_1, \ldots, b^{i_k}_1), \ldots,\ f(b^{i_1}_n, \ldots, b^{i_k}_n)\right)$$

or

$$\left(1,\ f(b^{i_1}_1, \ldots, b^{i_k}_1), \ldots,\ f(b^{i_1}_n, \ldots, b^{i_k}_n)\right).$$

2. Let R and f be as in 1., but f not constant, and let $R' = \forall x\, R(x, \boldsymbol{y})$. If then $\boldsymbol{b}^{i_1}, \ldots, \boldsymbol{b}^{i_k} \in R'$, also $\left(f(b^{i_1}_1, \ldots, b^{i_k}_1), \ldots,\ f(b^{i_1}_n, \ldots, b^{i_k}_n)\right) \in R'$: Since f is not constant, we can chose values $a^{i_1}_0, \ldots, a^{i_k}_0 \in \{0,1\}$ such that $f(a^{i_1}_0, \ldots, a^{i_k}_0) = 0$, and $a^{i_1}_1, \ldots, a^{i_k}_1 \in \{0,1\}$ such that $f(a^{i_1}_1, \ldots, a^{i_k}_1) = 1$. Since for each $\boldsymbol{b}^i \in R'$ both $(0, \boldsymbol{b}^i) \in R$ and $(1, \boldsymbol{b}^i) \in R$, R also contains both

$$\left(f(a^{i_1}_0, \ldots, a^{i_k}_0),\ f(b^{i_1}_1, \ldots, b^{i_k}_1), \ldots,\ f(b^{i_1}_n, \ldots, b^{i_k}_n)\right) = \left(0,\ f(b^{i_1}_1, \ldots, b^{i_k}_1), \ldots,\ f(b^{i_1}_n, \ldots, b^{i_k}_n)\right)$$

and

$$\left(f(a^{i_1}_1, \ldots, a^{i_k}_1),\ f(b^{i_1}_1, \ldots, b^{i_k}_1), \ldots,\ f(b^{i_1}_n, \ldots, b^{i_k}_n)\right) = \left(1,\ f(b^{i_1}_1, \ldots, b^{i_k}_1), \ldots,\ f(b^{i_1}_n, \ldots, b^{i_k}_n)\right).$$

□

Corollary 3.2.4 *Any relation obtained by arbitrarily quantifying over some variables of a bijunctive (Horn, dual Horn, affine, IHSB−, IHSB+) relation is itself bijunctive (Horn, dual Horn, affine, IHSB−, IHSB+).*

We are now ready to prove the dichotomy theorem:

Proof of Theorem 3.2.1. 1. We first show that the structural properties proved for

$\mathrm{CNF_C}(\mathcal{S})$-formulas with safely tight sets $\mathcal{S}$ in Section 2.5 also hold for Q-$\mathrm{CNF_C}(\mathcal{S})$-formulas if $\mathcal{S}$ is Schaefer.

If $\mathcal{S}$ is bijunctive or Horn (dual Horn), any $\mathrm{CNF_C}(\mathcal{S})$-formula ϕ is itself bijunctive, resp. Horn (dual Horn), so by Corollary 3.2.4, this also holds for any Q-$\mathrm{CNF_C}(\mathcal{S})$-formula. Now since by Lemma 2.1.9 every bijunctive relation is safely componentwise bijunctive, and every Horn resp. dual Horn relation is safely OR-free resp. safely NAND-free, the structural properties stated in Lemma 2.5.1 and Lemma 2.5.4 apply to Q-$\mathrm{CNF_C}(\mathcal{S})$-formulas also, and the statement for the diameter follows as in Lemma 2.5.2 and Lemma 2.5.5.

The above reasoning does not work for affine sets $\mathcal{S}$ since for such $\mathcal{S}$, $\mathrm{CNF_C}(\mathcal{S})$-formula are not necessarily affine itself. But nevertheless, the relations obtained by arbitrarily quantifying over $\mathrm{CNF_C}(\mathcal{S})$-formulas are expressible as $\mathrm{CNF_C}(\mathcal{S}')$-formulas for some affine set $\mathcal{S}'$: Figure 4.2 of [Bau07] shows an algorithm to transform any Q-$\mathrm{CNF_C}(\mathcal{S})$ formula ϕ into an equivalent system of linear equations, i.e. conjunction of affine expressions. Now since by Lemma 2.1.9, affine relations are safely componentwise bijunctive, safely OR-free, and safely NAND-free, the structural properties from Lemma 2.5.1 and Lemma 2.5.4 apply to Q-$\mathrm{CNF_C}(\mathcal{S})$-formulas also for affine $\mathcal{S}$.

It remains to prove that ST-Q-$\mathrm{CONN_C}(\mathcal{S})$ is in P and Q-$\mathrm{CONN_C}(\mathcal{S})$ is in coNP. The algorithms given in the proofs of Lemma 2.5.2 and Lemma 2.5.5 for showing that ST-$\mathrm{CONN_C}(\mathcal{S})$ is in P and $\mathrm{CONN_C}(\mathcal{S})$ is in coNP are by following paths in the solution graph in a given direction. In doing so, the formula has to be evaluated for a certain vector at each step; for Q-$\mathrm{CNF_C}(\mathcal{S})$-formulas, this means assigning the free variables and then evaluating the fully quantified formula. Now by Theorem 6.1 in [Sch78], the evaluation problem for quantified formulas is in P, so that the algorithms from proofs of Lemma 2.5.2 and Lemma 2.5.5 can also be used to solve the problems for Q-$\mathrm{CNF_C}(\mathcal{S})$-formulas in polynomial time.

2. By Schaefer's "expressibility theorem" (Theorem 3.0 of [Sch78]), if $\mathcal{S}$ is not Schaefer, every Boolean relation is expressible from $\mathcal{S}$ by existentially quantifying over some $\mathrm{CNF_C}(\mathcal{S})$ formula, and thus the statements follow from Lemma 2.3.1 and Lemma 2.3.2. □

3.2.2 coNP-Completeness for Connectivity

It remains to determine the complexity of Q-$\mathrm{CONN_C}$ for Schaefer sets of relations. We begin by showing that with quantifiers, we can extend the coNP-complete class. Since by Lemma 2.7.9, connectivity for $\mathcal{S} = \{(x \vee \overline{y} \vee \overline{z}) \wedge (\overline{x} \vee z)\}$ is coNP-hard already for $\mathrm{CNF_C}(\mathcal{S})$-formulas, the following lemma shows that Q-$\mathrm{CONN_C}(\mathcal{S})$ is coNP-hard for all sets of Horn relations that are not IHSB−.

Lemma 3.2.5 *The relation $M = (x \vee \overline{y} \vee \overline{z}) \wedge (\overline{x} \vee z)$ is expressible as an existentially quantified* $\mathrm{CNF_C}(\{R\})$*-formula for every Horn relation R that is not IHSB−.*

Proof. We will use quantification only at the very end. As in the proof of Lemma 2.7.10, in the following numbered transformation steps, we use identification and substitution to obtain one of a few simple formulas from which we can then express M as an existentially quantified formula.

We again argue with formulas in normal form ν; let $\phi_0 = \nu(R)$. Since R is not IHSB−, ϕ_0 contains a multi-implication $c = x \vee \overline{y} \vee \overline{z}_1 \vee \cdots \vee \overline{z}_k$ $(k \geq 1)$.

1. *Identify* $z_1, \ldots, x_k$*, call the resulting variable* z*.*

This produces the clause $x \vee \overline{y} \vee \overline{z}$ from c. Since by simplicity condition (c), x was not implied by any set $U \subsetneq \{y, z_1, \ldots, x_k\}$, and by (e), no z_i was implied by y, and y was implied by no set $U \subseteq \{z_1, \ldots, x_k\}$, it follows that

(*) $x \notin \mathrm{Imp}(y)$, $x \notin \mathrm{Imp}(z)$, $z \notin \mathrm{Imp}(y)$, $y \notin \mathrm{Imp}(z)$.

In the following steps, we eliminate all variables other than x, y, z, s.t. (*) is maintained.

2. *Substitute 1 for every variable from* $\mathrm{Imp}(y) \cap \mathrm{Imp}(z)$*.*

By (*), the set of these variables cannot have implied x, y, or z, thus there can emerge no unit clause on x, y or z; since by simplicity condition (d) the set of the substituted variables contains no restraint set, ϕ cannot become unsatisfiable.

3. *Identify all remaining variables from* $\mathrm{Imp}(y) \setminus \{y\}$ *with* y*.*

Since none of these variables was implied by z, still $y \notin \mathrm{Imp}(z)$ and it is easy to see that the other conditions of (*) are also maintained.

4. *Identify all remaining variables from* $\mathrm{Imp}(z) \setminus \{z\}$ *with* z*.*

Analogous to step 3, (*) is maintained. Now $\mathrm{Imp}(y) \setminus \{y\}$ and $\mathrm{Imp}(z) \setminus \{z\}$ are empty, so the last step is easy:

5. *Identify all remaining variables other than* x, y, z *with* x*.*

The formula now contains only the variables x, y, z, and all clauses satisfy (*). If follows that all clauses besides c are from

$$\{z \vee \overline{x},\ z \vee \overline{x} \vee \overline{y},\ y \vee \overline{x},\ y \vee \overline{x} \vee \overline{z}\}.$$

Considering all possible combinations of these clauses, we find that the the formula is equivalent to $K = x \vee \overline{y} \vee \overline{z}$, $L = (x \vee \overline{y} \vee \overline{z}) \wedge (\overline{x} \vee \overline{y} \vee z)$, $M = (x \vee \overline{y} \vee \overline{z}) \wedge (\overline{x} \vee z)$, $M' = (x \vee \overline{y} \vee \overline{z}) \wedge (\overline{x} \vee y)$, $S = (x \vee \overline{y} \vee \overline{z}) \wedge (y \vee \overline{x}) \wedge (z \vee \overline{x})$ or $T = (x \vee \overline{y} \vee \overline{z}) \wedge (y \vee \overline{z} \vee \overline{x}) \wedge (z \vee \overline{x} \vee \overline{y})$.

We express M from M' by permutation, and L from S or T as

$$L = \exists w S(w, y, x) \wedge S(w, y, z) = \exists w T(w, y, x) \wedge T(w, y, z).$$

Finally, we express M from K or L as in the proof of Lemma 2.7.10. □

Corollary 3.2.6 *If a finite set* $\mathcal{S}$ *of relations is Schaefer, but not bijunctive, IHSB−, IHSB+ or affine,* Q-CONN$_\mathrm{C}(\mathcal{S})$ *is* coNP*-complete.*

Proof. By Theorem 3.2.1, the problem is in coNP. From the definitions we find that $\mathcal{S}$ must be Horn and contain at least one relation that is not IHSB−, or dual Horn and contain at least one relation that is not IHSB+. In the first case, coNP-hardness follows from Lemma 2.7.9 with Lemma 3.2.5. The second case is symmetric. □

3.2.3 Deciding Connectivity in Polynomial Time

We are now left with bijunctive, IHSB$-$ / IHSB$+$, and affine sets of relations. We will devise a polynomial-time algorithm for connectivity in each case.

Lemma 3.2.7 *If $\mathcal{S}$ is a set of IHSB$-$ or IHSB$+$ relations, there is a polynomial-time algorithm for* Q-Conn$_\mathrm{C}(\mathcal{S})$.

Proof. The algorithm is essentially a modified version of Gopalan et al.'s algorithm from the proof of Lemma 4.13 in [GKMP09].

Assume $\mathcal{S}$ is IHSB$-$; the IHSB$+$ case is symmetric. Let ϕ be any Q-CNF$_\mathrm{C}(S)$-formula.

Since any CNF$_\mathrm{C}(S)$-formula is itself IHSB$-$, also $[\phi]$ can be expressed as an IHSB$-$ formula ψ without quantifiers due to Corollary 3.2.4. So if we could transform ϕ into ψ in polynomial time, we could then simply use the constraint-projection algorithm from Lemma 2.6.5 to decide connectivity. However, quantifier-elimination can lead to an exponential increase of the formula size, even for Horn formulas [BB08], and this seems to apply to IHSB$-$ formulas also. Thus we need another strategy.

Fortunately, there at least exists a polynomial time algorithm to transform a quantified Horn formula into an equivalent Horn formula with only existential quantifiers, described by Bubeck et al. in Definition 8 of [BB08]; we apply this algorithm to ϕ to obtain an equivalent formula $\phi^\exists$ with only existential quantifiers.

Next we assign all variables in positive unit clauses (and remove the corresponding quantifiers in the case of bound variables) to obtain a connectivity-equivalent formula $\phi^{\exists\prime}$ without positive unit clauses; let $\phi^{\exists\,\prime} = \exists x_1 \cdots \exists x_m \phi_0(x_1, \dots, x_n)$, where ϕ_0 contains no quantifiers.

We show that

(*) $\phi^{\exists\prime}$ (and thus ϕ) is disconnected iff

  (1) there are two free variables x and y of $\phi^{\exists\prime}$ s.t. $x \in \mathrm{Imp}_{\phi_0}(y)$[2], $y \in \mathrm{Imp}_{\phi_0}(x)$, and $\mathrm{Imp}_{\phi_0}(x)$ contains no restraint set of ϕ_0.

It is easy to see that condition (1) can be checked in polynomial time as follows:

For every free variable x of $\phi^{\prime\exists}$:

  If $\mathrm{Imp}_{\phi_0}(x)$ contains no restraint set of ϕ_0,

    For each free variable y from $\mathrm{Imp}_{\phi_0}(x)$:

      If $x \in \mathrm{Imp}_{\phi_0}(y)$, return "disconnected".

Return "connected".

To prove (*), consider the formula $\phi^\sim$ we would obtain from $\phi^{\exists\prime}$ by eliminating all quantifiers, and then transforming into conjunctive normal form. It is clear that also $\phi^\sim$ contains no positive unit-clauses. Further, $\phi^\sim$ is again IHSB$-$ by Corollary 3.2.4, and thus Horn. So by Corollary 2.7.6, $\phi^\sim$ (and therefore ϕ) is disconnected iff

[2] Imp_ψ denotes implication in formula ψ

(2) $\phi^{\sim}$ has a non-empty maximal self-implicating set containing no restraint set.

Since $\phi^{\sim}$ is IHSB$-$, it contains no multi-implication clauses; therefore, in the hypergraph-representation, all implication clauses of $\phi^{\sim}$ correspond to simple edges, not hyperedges. Thus, the implication clauses of $\phi^{\sim}$ can be represented as an ordinary digraph, and implication corresponds to reachability in that digraph. With this, it is easy to see that (2) is equivalent to

(3) in $\phi^{\sim}$, there are two variables x and y with $x \in$Imp(y), $y \in$Imp(x), and s.t. Imp(x) contains no restraint set.

It remains to show that (1) is equivalent to (3). The proof will be by induction. Therefor first note that $\phi^{\sim}$ can be obtained from $\phi^{\exists'}$ by eliminating the quantifiers one by one, and always writing the resulting formula in conjunctive normal form. It follows that if we define $\phi_k = \mathrm{CNF}(\phi_{k-1}[x_k/0] \vee \phi_{k-1}[x_k/1])$, where CNF$(\psi)$ denotes a CNF-formula equivalent to ψ, then $\phi^{\sim} = \phi_m$. Let

$$\phi_k = c_1 \wedge \cdots \wedge c_p \wedge d_1 \wedge \cdots \wedge d_q,$$

where $c_1, \ldots, c_p$ resp. $d_1, \ldots, d_q$ are the clauses containing x_k resp. not containing x_k. Then

$$\phi_{k-1}[x_k/0] \vee \phi_{k-1}[x_k/1] \equiv \left(\bigwedge_{i,j} c_i[y_k/0] \vee c_j[y_k/1] \right) \wedge d_1 \wedge \cdots \wedge d_q.$$

The clauses $c_1, \ldots, c_p$ are of the form $\overline{x}_i \vee x_k$, $x_i \vee \overline{x_k}$, or $\overline{x}_k \vee \overline{x}_{i_1} \vee \cdots \vee \overline{x}_{i_r}$; considering all combinations to the disjunctions $c_i[y_k/0] \vee c_j[y_k/1]$, and discarding tautological clauses, it is then easy to see that ϕ_k may[3] consist of exactly the following clauses:

1. $d_1, \ldots d_q$,
2. for each pair of clauses $\overline{x}_i \vee x_k$ and $x_j \vee \overline{x_k}$ of ϕ_{k-1} with $x_i \neq x_j$, we have $\overline{x}_i \vee x_j$ in ϕ_k,
3. for each pair of clauses $\overline{x}_i \vee x_k$ and $\overline{x}_k \vee \overline{x}_{i_1} \vee \cdots \vee \overline{x}_{i_r}$ of ϕ_{k-1}, we have $\overline{x}_i \vee \overline{x}_{i_1} \vee \cdots \vee \overline{x}_{i_r}$ in ϕ_k.

We can now show that for any two variables x and y with $x \neq x_k$ and $y \neq x_k$,

(a) $x \in \mathrm{Imp}_{\phi_k}(y)$ iff $x \in \mathrm{Imp}_{\phi_{k-1}}(y)$,

Proof: "$\Longleftarrow$": If $x \in \mathrm{Imp}_{\phi_{k-1}}(y)$, there was a chain of clauses

$$\overline{y} \vee x_{w_1},\ \overline{x}_{w_1} \vee x_{w_2},\ \ldots,\ \overline{x}_{w_m} \vee x$$

in ϕ_{k-1}; now if $x_k \notin \{x_{w_1}, \ldots, x_{w_k}\}$, all clauses of the chain are also in ϕ_k by 1., else a clause "bridging" x_k is added by 2.

"$\Longrightarrow$": This is clear since all implication clauses of ϕ_k are entailed by ϕ_{k-1}.

(b) $\mathrm{Imp}_{\phi_k}(x)$ contains a restraint set U' of ϕ_k iff $\mathrm{Imp}_{\phi_{k-1}}(x)$ contains a restraint set U of ϕ_{k-1}.

[3] a CNF representation is not unique

Proof: "$\Longleftarrow$": If U did not contain x_k, U is also a restraint set of ϕ_k, and by 1., $U \subseteq \mathrm{Imp}_{\phi_k}(x)$. Otherwise, $U = \{x_k, x_{i_1}, \ldots, x_{i_r}\}$ with each $x_{i_j} \in \mathrm{Imp}_{\phi_{k-1}}(x)$, and there was some clause $\overline{x}_l \vee x_k$ with $x_l \in \mathrm{Imp}_{\phi_{k-1}}(x)$, but then ϕ_k contains the clause $\overline{x}_l \vee \overline{x}_{i_1} \vee \cdots \vee \overline{x}_{i_r}$ by 3., and since also all $x_{i_j} \in \mathrm{Imp}_{\phi_k}(x)$ and $x_l \in \mathrm{Imp}_{\phi_k}(x)$ by 1., $U' = \{x_l, x_{i_1}, \ldots, x_{i_r}\} \subseteq \mathrm{Imp}_{\phi_k}(x)$.

"$\Longrightarrow$": If U' corresponds to one of the clauses $d_1, \ldots d_q$, it was also a restraint set of ϕ_{k-1}, and $U' \subseteq \mathrm{Imp}_{\phi_{k-1}}(x)$ by 1. Otherwise, it corresponds to a clause $\overline{x}_i \vee \overline{x}_{i_1} \vee \cdots \vee \overline{x}_{i_r}$ from 3.; but then ϕ_{k-1} must have contained the clauses $\overline{x}_i \vee x_k$ and $\overline{x}_k \vee \overline{x}_{i_1} \vee \cdots \vee \overline{x}_{i_r}$, and $U = \{x_k, x_{i_1}, \ldots, x_{i_r}\} \subseteq \mathrm{Imp}_{\phi_{k-1}}(x)$ by 1.

Now (1)$\Longleftrightarrow$(3) follows from (a) and (b) by induction. □

We can reduce the bijunctive case to the IHSB− one:

Lemma 3.2.8 *If $\mathcal{S}$ is a bijunctive set of relations,* Q-CONN$_\mathrm{C}(\mathcal{S})$ *is in* P*.*

Proof. Makino, Tamaki and Yamamoto show in [MTY07] below Proposition 2 that any bijunctive formula can be transformed in a connectivity-equivalent Horn 2-CNF formula by "renaming" variables: We can calculate a solution $\boldsymbol{a}$ in linear time [APT79] (w.l.o.g. we may assume that a solution exists) and then take

$$\psi(\boldsymbol{x}) = \phi(x_1 \oplus a_1, x_2 \oplus a_2, \ldots).$$

Now ψ is clearly Horn since $\psi(0, \ldots 0) = 1$, and the connectivity is retained since

$$|(x_1 \oplus a_1, x_2 \oplus a_2, \ldots) - (y_1 \oplus a_1, y_2 \oplus a_2, \ldots)| = |(x_1, x_2, \ldots) - (y_1, y_2, \ldots)|.$$

Since any CNF$_\mathrm{C}(\mathcal{S})$ formula ϕ is itself bijunctive, given $Q_1 y_1 \cdots Q_m y_m \phi(\boldsymbol{x}, \boldsymbol{y})$, we can instead take $Q_1 y_1 \cdots Q_m y_m \psi(\boldsymbol{x}, \boldsymbol{y})$, where ψ is the Horn 2-CNF formula obtained from ϕ as described, and then apply the algorithm for the IHSB− case, since any Horn 2-CNF formula is also IHSB−. □

Finally, for affine sets of relations, we again make use of the quantifier-elimination algorithm from [Bau07], and then use an algorithm from Gopalan et al.:

Lemma 3.2.9 *If $\mathcal{S}$ is an affine set of relations,* Q-CONN$_\mathrm{C}(\mathcal{S})$ *is in* P*.*

Proof. We can use the polynomial-time algorithm from Figure 4.2 in [Bau07] to transform any quantified CNF$_\mathrm{C}(\mathcal{S})$ formula into an equivalent one without any quantifiers, and then apply the algorithm from the proof of Lemma 4.10 of [GKMP09]. Note that although the clause-size of the formulas produced by the algorithm from [Bau07] is unbounded (with regard to $\mathcal{S}$), the algorithm from [GKMP09] decides connectivity for these formulas in polynomial time, as is easy to see. □

4 Connectivity of Nested Formulas and Circuits

We now turn to a quite different type of representation for Boolean relations. First observe that we can naturally identify Boolean relations with *Boolean functions*, i.e. functions $f : \{0,1\}^n \to \{0,1\}$. To build B-formulas, we again use a fixed finite set of Boolean relations, resp. functions, as source material, but instead of connecting them with $\wedge$'s, we now allow inserting them into each other arbitrarily.

In a B-formula, if some function is used repeatedly with the same arguments, it has to be duplicated, what can make the formula large and evaluation slow. This is remedied by B-circuits, that allow to use the result of a function any number of times, which can lead to exponential savings of space and time.

4.1 Preliminaries: *B*-Circuits, *B*-Formulas, and Post's Lattice

We begin with formal definitions of B-circuits and B-formulas. Let B be a finite set of Boolean functions.

Definition 4.1.1 A *B-circuit* $\mathcal{C}$ with input variables $x_1, \ldots, x_n$ is a directed acyclic graph, augmented as follows: Each node (here also called *gate*) with indegree 0 is labeled with an x_i or a 0-ary function from B, each node with indegree $k > 0$ is labeled with a k-ary function from B. The edges (here also called *wires*) pointing into a gate are ordered. One node is designated the output gate. Given values $a_1, \ldots, a_n \in \{0,1\}$ to $x_1, \ldots, x_n$, $\mathcal{C}$ computes an n-ary function $f_{\mathcal{C}}$ as follows: A gate v labeled with a variable x_i returns a_i, a gate v labeled with a function f computes the value $f(b_1, \ldots, b_k)$, where $b_1, \ldots, b_k$ are the values computed by the predecessor gates of v, ordered according to the order of the wires.

For a detailed introduction to Boolean circuits and circuit complexity, see e.g. [Vol99].

Definition 4.1.2 A *B-formula* is defined inductively: A variable x is a B-formula. If $\phi_1, \ldots, \phi_m$ are B-formulas, and f is an n-ary function from B, then $f(\phi_1, \ldots, \phi_n)$ is a B-formula. In turn, any B-formula defines a Boolean function in the obvious way, and we will identify B-formulas with the function they define.

It is easy to see that the functions computable by a B-circuit, as well as the functions definable by a B-formula, are exactly those that can be obtained from B by *superposition*, together with all projections [BCRV03]. By superposition, we mean substitution (that is, composition of functions), permutation and identification of variables, and introduction of *fictive variables* (variables on which the value of the function does not depend). This class of functions is denoted $[B]$. B is *closed* (or said to be a *clone*) if $[B] = B$. A *base* of a clone F is any set B with $[B] = F$.

Already in the early 1920s, Emil Post extensively studied Boolean functions [Pos41]. He identified all closed classes, found a finite base for each of them, and detected their inclusion structure: The closed classes form a lattice, called *Post's lattice*, depicted in Figure 4.1.1. We do not use Post's original names for the closed classes, but the modern terminology developed by Reith and Wagner in [RW99]; the layout of the lattice is also from [RW99].

The following clones are defined by properties of the functions they contain, all other ones are intersections of these. Let f be an n-ary Boolean function.

- BF is the class of all Boolean functions.
- R_0 (R_1) is the class of all 0-reproducing (1-reproducing) functions,
  f is *c-reproducing*, if $f(c, \dots, c) = c$, where $c \in \{0, 1\}$.
- M is is the class of all monotone functions,
  f is *monotone*, if $a_1 \leq b_1, \dots, a_n \leq b_n$ implies $f(a_1, \dots, a_n) \leq f(b_1, \dots, b_n)$.
- D is the class of all self-dual functions,
  f is *self-dual*, if $f(x_1, \dots, x_n) = \overline{f(\overline{x_1}, \dots, \overline{x_n})}$.
- L is the class of all affine (on *linear*) functions,
  f is *affine*, if $f(x_1, \dots, x_n) = x_{i_1} \oplus \cdots \oplus x_{i_m} \oplus c$ with $i_1, \dots, i_m \in \{1, \dots, n\}$ and $c \in \{0, 1\}$.
- S_0 (S_1) is the class of all 0-separating (1-separating) functions,
  f is *c-separating*, if there exists an $i \in \{1, \dots, n\}$ s.t. $a_i = c$ for all $\boldsymbol{a} \in f^{-1}(c)$, where $c \in \{0, 1\}$.
- S_0^m (S_1^m) is the class of all functions 0-separating (1-separating) of degree m,
  f is *c-separating of degree m*, if for all $U \subseteq f^{-1}(c)$ of size $|U| = m$ there exists an $i \in \{1, \dots, n\}$ s.t. $a_i = c$ for all $\boldsymbol{a} \in U$ ($c \in \{0, 1\}$, $m \geq 2$).

The definitions and bases of all classes are given in Table 4.1. For an introduction to Post's lattice and further references see e.g. [BCRV03].

The complexity of numerous problems for B-circuits and B-formulas has been classified by the types of functions allowed in B with help of Post's lattice (see e.g. [RW00, Sch07]), starting with satisfiability: Analogously to Schaefer's dichotomy for CNF($\mathcal{S}$)-formulas from 1978, Harry R. Lewis shortly thereafter found a dichotomy for B-formulas [Lew79]: If $[B]$ contains the function $x \wedge \overline{y}$, SAT is NP-complete, else it is in P.

While for B-circuits the complexity of every decision problem solely depends on $[B]$ (up to AC^0 isomorphisms), for B-formulas this need not be the case (though it usually is, as for satisfiability and our connectivity problems, as we will see): The transformation of a B-formula into a B'-formula might require an exponential increase in the formula size even if $[B] = [B']$, as the B'-representation of some function from B may need to use some input variable more than once [Tho12]. For example, let $h(x, y) = x \wedge \overline{y}$; then (x∧y) $\in [\{h\}]$ since $x \wedge y = h(x, h(x, y))$, but it is easy to see that there is no shorter $\{h\}$-representation of $x \wedge y$.

We denote the *st*-connectivity and connectivity problems for B-formulas by ST-BF-CONN(B) and BF-CONN(B), respectively, and the problems for circuits by ST-CIRC-CONN(B) resp. CIRC-CONN(B).

Also for a Boolean function f, we will speak of the solution graph $G(f)$ and denote the shortest-path distance in $G(f)$ by d_f.

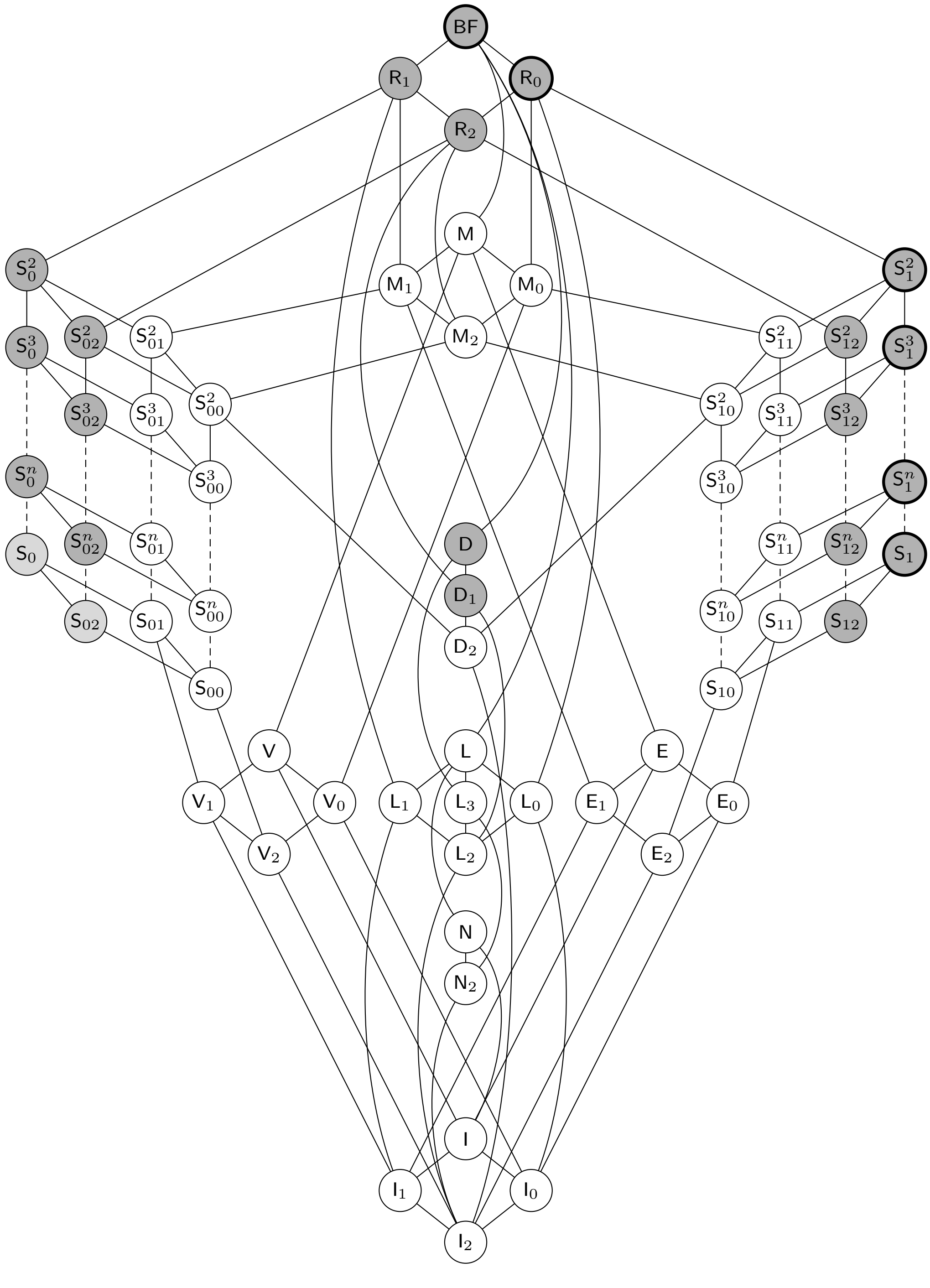


Figure 4.1.1 Post's lattice with our results.
The classes on the hard side of the dichotomy for the connectivity problems and the diameter are shaded; the light shaded ones are on the hard side only for formulas with quantifiers.

For comparison, the classes for which SAT (without quantifiers) is NP-complete are circled bold.

Class	Definition	Base
BF	All Boolean functions	$\{x \wedge y, \neg x\}$
R_0	$\{f \in \mathsf{BF} \mid f$ is 0-reproducing$\}$	$\{x \wedge y, x \oplus y\}$
R_1	$\{f \in \mathsf{BF} \mid f$ is 1-reproducing$\}$	$\{x \vee y, x \leftrightarrow y\}$
R_2	$\mathsf{R}_0 \cap \mathsf{R}_1$	$\{x \vee y, x \wedge (y \leftrightarrow z)\}$
M	$\{f \in \mathsf{BF} \mid f$ is monotone$\}$	$\{x \wedge y, x \vee y, 0, 1\}$
M_0	$\mathsf{M} \cap \mathsf{R}_0$	$\{x \wedge y, x \vee y, 0\}$
M_1	$\mathsf{M} \cap \mathsf{R}_1$	$\{x \wedge y, x \vee y, 1\}$
M_2	$\mathsf{M} \cap \mathsf{R}_2$	$\{x \wedge y, x \vee y\}$
S_0	$\{f \in \mathsf{BF} \mid f$ is 0-separating$\}$	$\{x \rightarrow y\}$
S_0^n	$\{f \in \mathsf{BF} \mid f$ is 0-separating of degree $n\}$	$\{x \rightarrow y, \mathrm{dual}(\mathrm{T}_n^{n+1})\}$
S_1	$\{f \in \mathsf{BF} \mid f$ is 1-separating$\}$	$\{x \nrightarrow y\}$
S_1^n	$\{f \in \mathsf{BF} \mid f$ is 1-separating of degree $n\}$	$\{x \nrightarrow y, \mathrm{T}_n^{n+1}\}$
S_{02}^n	$\mathsf{S}_0^n \cap \mathsf{R}_2$	$\{x \vee (y \wedge \neg z), \mathrm{dual}(\mathrm{T}_n^{n+1})\}$
S_{02}	$\mathsf{S}_0 \cap \mathsf{R}_2$	$\{x \vee (y \wedge \neg z)\}$
S_{01}^n	$\mathsf{S}_0^n \cap \mathsf{M}$	$\{\mathrm{dual}(\mathrm{T}_n^{n+1}), 1\}$
S_{01}	$\mathsf{S}_0 \cap \mathsf{M}$	$\{x \vee (y \wedge z), 1\}$
S_{00}^n	$\mathsf{S}_0^n \cap \mathsf{R}_2 \cap \mathsf{M}$	$\{x \vee (y \wedge z), \mathrm{dual}(\mathrm{T}_n^{n+1})\}$
S_{00}	$\mathsf{S}_0 \cap \mathsf{R}_2 \cap \mathsf{M}$	$\{x \vee (y \wedge z)\}$
S_{12}^n	$\mathsf{S}_1^n \cap \mathsf{R}_2$	$\{x \wedge (y \vee \neg z), \mathrm{T}_n^{n+1}\}$
S_{12}	$\mathsf{S}_1 \cap \mathsf{R}_2$	$\{x \wedge (y \vee \neg z)\}$
S_{11}^n	$\mathsf{S}_1^n \cap \mathsf{M}$	$\{\mathrm{T}_n^{n+1}, 0\}$
S_{11}	$\mathsf{S}_1 \cap \mathsf{M}$	$\{x \wedge (y \vee z), 0\}$
S_{10}^n	$\mathsf{S}_1^n \cap \mathsf{R}_2 \cap \mathsf{M}$	$\{x \wedge (y \vee z), \mathrm{T}_n^{n+1}\}$
S_{10}	$\mathsf{S}_1 \cap \mathsf{R}_2 \cap \mathsf{M}$	$\{x \wedge (y \vee z)\}$
D	$\{f \in \mathsf{BF} \mid f$ is self-dual$\}$	$\{\mathrm{maj}(x, \neg y, \neg z)\}$
D_1	$\mathsf{D} \cap \mathsf{R}_2$	$\{\mathrm{maj}(x, y, \neg z)\}$
D_2	$\mathsf{D} \cap \mathsf{M}$	$\{\mathrm{maj}(x, y, z)\}$
L	$\{f \in \mathsf{BF} \mid f$ is linear$\}$	$\{x \oplus y, 1\}$
L_0	$\mathsf{L} \cap \mathsf{R}_0$	$\{x \oplus y\}$
L_1	$\mathsf{L} \cap \mathsf{R}_1$	$\{x \leftrightarrow y\}$
L_2	$\mathsf{L} \cap \mathsf{R}_2$	$\{x \oplus y \oplus z\}$
L_3	$\mathsf{L} \cap \mathsf{D}$	$\{x \oplus y \oplus z \oplus 1\}$
E	$\{f \in \mathsf{BF} \mid f$ is constant or a conjunction$\}$	$\{x \wedge y, 0, 1\}$
E_0	$\mathsf{E} \cap \mathsf{R}_0$	$\{x \wedge y, 0\}$
E_1	$\mathsf{E} \cap \mathsf{R}_1$	$\{x \wedge y, 1\}$
E_2	$\mathsf{E} \cap \mathsf{R}_2$	$\{x \wedge y\}$
V	$\{f \in \mathsf{BF} \mid f$ is constant or a disjunction$\}$	$\{x \vee y, 0, 1\}$
V_0	$\mathsf{V} \cap \mathsf{R}_0$	$\{x \vee y, 0\}$
V_1	$\mathsf{V} \cap \mathsf{R}_1$	$\{x \vee y, 1\}$
V_2	$\mathsf{V} \cap \mathsf{R}_2$	$\{x \vee y\}$
N	$\{f \in \mathsf{BF} \mid f$ is essentially unary$\}$	$\{\neg x, 0, 1\}$
N_2	$\mathsf{N} \cap \mathsf{D}$	$\{\neg x\}$
I	$\{f \in \mathsf{BF} \mid f$ is constant or a projection$\}$	$\{x, 0, 1\}$
I_0	$\mathsf{I} \cap \mathsf{R}_0$	$\{x, 0\}$
I_1	$\mathsf{I} \cap \mathsf{R}_1$	$\{x, 1\}$
I_2	$\mathsf{I} \cap \mathsf{R}_2$	$\{x\}$

Table 4.1 List of all closed classes of Boolean functions with definitions and bases. T_k^n denotes the threshold function, i.e., $T_k^n(x_1, \ldots, x_n) = 1 \iff \sum_{i=1}^n x_i \geq k$, $(\mathrm{dual}(f))(x_1, \ldots, x_n) = \overline{f(\overline{x_1}, \ldots, \overline{x_n})}$)

4.2 Results

The following two theorems give complete classifications up to polynomial-time isomorphisms. See also Figure 4.1.1.

Theorem 4.2.1 *Let B be a finite set of Boolean functions.*

1. *If $B \subseteq \mathsf{M}$, $B \subseteq \mathsf{L}$, or $B \subseteq \mathsf{S}_0$, then*
   a) st-Circ-Conn(B) *and* Circ-Conn(B) *are in* P*,*
   b) st-BF-Conn(B) *and* BF-Conn(B) *are in* P*,*
   c) *for every B-formula ϕ, the diameter of $G(\phi)$ is linear in the number of variables.*
2. *Otherwise,*
   a) st-Circ-Conn(B) *and* Circ-Conn(B) *are* PSPACE*-complete,*
   b) st-BF-Conn(B) *and* BF-Conn(B) *are* PSPACE*-complete,*
   c) *there are B-formulas ϕ such that the diameter of $G(\phi)$ is exponential in the number of variables.*

The proof follows from the lemmas in the next two sections.

4.3 The Easy Side of the Dichotomy

We will often use the following proposition to relate the complexity of B-formulas and B-circuits:

Proposition 4.3.1 *Every B-formula ϕ can be transformed into an equivalent B-circuit $\mathcal{C}$ in polynomial time.*

Proof. Any B-formula is equivalent to a special B-circuit where all function-gates have outdegree at most one: For every variable x of ϕ and for every occurrence of a function f in ϕ there is a gate in $\mathcal{C}$, labeled with x resp. f. It is clear how to connect the gates. □

Lemma 4.3.2 *If $B \subseteq \mathsf{M}$, the solution graph of any n-ary function $f \in [B]$ is connected, and $d_f(\boldsymbol{a}, \boldsymbol{b}) = |\boldsymbol{a} - \boldsymbol{b}| \leq n$ for any two solutions $\boldsymbol{a}$ and $\boldsymbol{b}$.*

Proof. Table 4.1 shows that f is monotone in this case. Thus, either $f = 0$, or $(1, \ldots, 1)$ must be a solution, and every other solution $\boldsymbol{a}$ is connected to $(1, \ldots, 1)$ in $G(f)$ since $(1, \ldots, 1)$ can be reached by flipping the variables assigned 0 in $\boldsymbol{a}$ one at a time to 1. Further, if $\boldsymbol{a}$ and $\boldsymbol{b}$ are solutions, $\boldsymbol{b}$ can be reached from $\boldsymbol{a}$ in $|\boldsymbol{a} - \boldsymbol{b}|$ steps by first flipping all variables that are assigned 0 in $\boldsymbol{a}$ and 1 in $\boldsymbol{b}$, and then flipping all variables that are assigned 1 in $\boldsymbol{a}$ and 0 in $\boldsymbol{b}$. □

Lemma 4.3.3 *If $B \subseteq \mathsf{S}_0$, the solution graph of any function $f \in [B]$ is connected, and $d_f(\boldsymbol{a}, \boldsymbol{b}) \leq |\boldsymbol{a} - \boldsymbol{b}| + 2$ for any two solutions $\boldsymbol{a}$ and $\boldsymbol{b}$.*

Proof. Since f is 0-separating, there is an i such that $a_i = 0$ for every vector $\boldsymbol{a}$ with $f(\boldsymbol{a}) = 0$, thus every $\boldsymbol{b}$ with $b_i = 1$ is a solution. It follows that every solution $\boldsymbol{t}$ can be reached from any solution $\boldsymbol{s}$ in at most $|\boldsymbol{s}-\boldsymbol{t}|+2$ steps by first flipping the i-th variable from 0 to 1 if necessary, then flipping all other variables in which $\boldsymbol{s}$ and $\boldsymbol{t}$ differ, and finally flipping back the i-th variable if necessary. □

Lemma 4.3.4 *If $B \subseteq \mathsf{L}$,*

1. st-Circ-Conn(B) *and* Circ-Conn(B) *are in P,*
2. st-BF-Conn(B) *and* BF-Conn(B) *are in P,*
3. *for any function $f \in [B]$, $d_f(\boldsymbol{a}, \boldsymbol{b}) = |\boldsymbol{a} - \boldsymbol{b}|$ for any two solutions $\boldsymbol{a}$ and $\boldsymbol{b}$ that lie in the same connected component of $G(\phi)$.*

Proof. Since every function $f \in \mathsf{L}$ is linear, $f(x_1, \ldots, x_n) = x_{i_1} \oplus \ldots \oplus x_{i_m} \oplus c$, and any two solutions $\boldsymbol{s}$ and $\boldsymbol{t}$ are connected iff they differ only in fictive variables: If $\boldsymbol{s}$ and $\boldsymbol{t}$ differ in at least one non-fictive variable (i.e., an $x_i \in \{x_{i_1}, \ldots, x_{i_m}\}$), to reach $\mathbf{t}$ from $\mathbf{s}$, x_i must be flipped eventually, but for every solution $\boldsymbol{a}$, any vector $\boldsymbol{b}$ that differs from $\boldsymbol{a}$ in exactly one non-fictive variable is no solution. If $\boldsymbol{s}$ and $\boldsymbol{t}$ differ only in fictive variables, $\boldsymbol{t}$ can be reached from $\boldsymbol{s}$ in $|\boldsymbol{s} - \boldsymbol{t}|$ steps by flipping one by one the variables in which they differ.

Since $\{x \oplus y, 1\}$ is a base of L, every B-circuit $\mathcal{C}$ can be transformed in polynomial time into an equivalent $\{x \oplus y, 1\}$-circuit $\mathcal{C}'$ by replacing each gate of $\mathcal{C}$ with an equivalent $\{x \oplus y, 1\}$-circuit. Now one can decide in polynomial time whether a variable x_i is fictive by checking for $\mathcal{C}'$ whether the number of "backward paths" from the output gate to gates labeled with x_i is odd, so st-Circ-Conn(B) is in P.

$G(\mathcal{C})$ is connected iff at most one variable is non-fictive, thus Circ-Conn(B) is in P.

By Proposition 4.3.1, st-BF-Conn(B) and BF-Conn(B) are in P also. □

This completes the proof of the easy side of the dichotomy.

4.4 The Hard Side of the Dichotomy

Clearly, we can transfer the upper bounds for general CNF-formulas to B-formulas and B-circuits:

Proposition 4.4.1 st-Circ-Conn(B) *and* Circ-Conn(B)*, as well as* st-BF-Conn(B) *and* BF-Conn(B)*, are in* PSPACE *for any finite set B of Boolean functions.*

Proof. This follows as in Lemma 3.6 of [GKMP09] (see Lemma 2.3.1). □

All hardness proofs will be by reductions from the problems for 1-reproducing 3-CNF-formulas, which are PSPACE-complete by the following proposition.

Proposition 4.4.2 *For 1-reproducing 3-CNF-formulas, the problems* st-Conn *and* Conn *are* PSPACE*-hard.*

Proof. In the PSPACE-hardness proof for general 3-CNF-formulas (Lemma 3.6 of [GKMP09], see Lemma 2.3.1), two satisfying assignments $\boldsymbol{s}$ and $\boldsymbol{t}$ to the constructed formula ϕ are known, so we can construct a connectivity-equivalent 1-reproducing 3-CNF-formula ψ, e.g. as $\psi(\boldsymbol{x}) = \phi(x_1 \oplus s_1 \oplus 1, \ldots, x_n \oplus s_n \oplus 1)$, and then check connectivity for ψ instead of ϕ. □

An inspection of Post's lattice shows that if $B \nsubseteq \mathsf{M}$, $B \nsubseteq \mathsf{L}$, and $B \nsubseteq \mathsf{S}_0$, then $[B] \supseteq \mathsf{S}_{12}$, $[B] \supseteq \mathsf{D}_1$, or $[B] \supseteq \mathsf{S}_{02}^k \, \forall k \geq 2$, so we have to prove PSPACE-completeness and show the existence of B-formulas with an exponential diameter in these cases.

Definition 4.4.3 We write $\boldsymbol{x} = \boldsymbol{c}$ or $\boldsymbol{x} = c_1 \cdots c_n$ for $(x_1 = c_1) \wedge \cdots \wedge (x_n = c_n)$, where $\boldsymbol{c} = (c_1, \ldots, c_n)$ is a vector of constants; e.g.. $\boldsymbol{x} = \boldsymbol{0}$ means $\overline{x}_1 \wedge \cdots \wedge \overline{x}_n$, and $\boldsymbol{x} = 101$ means $x_1 \wedge \overline{x}_2 \wedge x_3$. Further, we use $\boldsymbol{x} \in \{\boldsymbol{a}, \boldsymbol{b}, \ldots\}$ for $(\boldsymbol{x} = \boldsymbol{a}) \vee (\boldsymbol{x} = \boldsymbol{b}) \vee \ldots$. Also, we write $\psi(\overline{\boldsymbol{x}})$ for $\psi(\overline{x}_1, \ldots, \overline{x}_n)$. If we have two vectors of Boolean values $\boldsymbol{a}$ and $\boldsymbol{b}$ of length n and m resp., we write $\boldsymbol{a} \cdot \boldsymbol{b}$ for their concatenation $(a_1, \ldots, a_n, b_1, \ldots b_m)$.

Lemma 4.4.4 *If* $[B] \supseteq \mathsf{S}_{12}$*,*

1. st-BF-Conn(B) *and* BF-Conn(B) *are* PSPACE*-complete,*
2. st-Circ-Conn(B) *and* Circ-Conn(B) *are* PSPACE*-complete,*
3. *for* $n \geq 3$*, there is an* n*-ary function* $f \in [B]$ *with diameter of at least* $2^{\lfloor \frac{n-1}{2} \rfloor}$*.*

Proof. 1. We reduce the problems for 1-reproducing 3-CNF-formulas to the ones for B-formulas: We map a 1-reproducing 3-CNF-formula ϕ and two solutions $\boldsymbol{s}$ and $\boldsymbol{t}$ of ϕ to a B-formula ϕ' and two solutions $\boldsymbol{s'}$ and $\boldsymbol{t'}$ of ϕ' such that $\boldsymbol{s'}$ and $\boldsymbol{t'}$ are connected in $G(\phi')$ iff $\boldsymbol{s}$ and $\boldsymbol{t}$ are connected in $G(\phi)$, and such that $G(\phi')$ is connected iff $G(\phi)$ is connected.

While the construction of ϕ' is quite easy for this lemma, the construction for the next two lemmas is analogous but more intricate, so we proceed carefully in two steps, which we will adapt in the next two proofs: In the first step, we give a transformation T that transforms any 1-reproducing formula ψ into a connectivity-equivalent formula $T_\psi \in \mathsf{S}_{12}$ built from the standard connectives. Since $\mathsf{S}_{12} \subseteq [B]$, we can express T_ψ as a B-formula T_ψ^*. Now if we would apply T to ϕ directly, we would know that T_ϕ can be expressed as a B-formula. However, this could lead to an exponential increase in the formula size (see Section 4.1), so we have to show how to construct the B-formula in polynomial time. For this, in the second step, we construct a B-formula ϕ' directly from ϕ (by applying T to the clauses and the $\wedge$'s individually), and then show that ϕ' is equivalent to T_ϕ; thus we know that ϕ' is connectivity-equivalent to ϕ.

Step 1. From Table 4.1, we find that $\mathsf{S}_{12} = \mathsf{S}_1 \cap \mathsf{R}_2 = \mathsf{S}_1 \cap \mathsf{R}_0 \cap \mathsf{R}_1$, so we have to make sure that T_ψ is 1-seperating, 0-reproducing, and 1-reproducing. Let

$$T_\psi = \psi \wedge y,$$

where y is a new variable.

All solutions $\boldsymbol{a}$ of $T_\psi(\boldsymbol{x}, y)$ have $a_{n+1} = 1$, so T_ψ is 1-seperating and 0-reproducing; also, T_ψ is still 1-reproducing. Further, for any two solutions $\boldsymbol{s}$ and $\boldsymbol{t}$ of $\psi(\boldsymbol{x})$, $\boldsymbol{s'} = \boldsymbol{s} \cdot 1$ and $\boldsymbol{t'} = \boldsymbol{t} \cdot 1$ are solutions of $T_\psi(\boldsymbol{x}, y)$, and it is easy to see that they are connected in $G(T_\psi)$ iff $\boldsymbol{s}$ and $\boldsymbol{t}$ are connected in $G(\psi)$, and that $G(T_\psi)$ is connected iff $G(\psi)$ is connected.

Step 2. The idea is to parenthesize the conjunctions of ϕ such that we get a tree of $\wedge$'s of depth logarithmic in the size of ϕ, and then to replace each clause and each $\wedge$ with an equivalent B-formula. This can increase the formula size by only a polynomial in the original size even if the B-formula equivalent to $\wedge$ uses some input variable more than once.

Let $\phi = C_1 \wedge \cdots \wedge C_n$ be a 1-reproducing 3-CNF-formula. Since ϕ is 1-reproducing, every clause C_i of ϕ is itself 1-reproducing, and we can express T_{C_i} through a B-formula $T^*_{C_i}$. Also, we can express $T_{u\wedge v}$ through a B-formula $T^*_{u\wedge v}$ since $\wedge$ is 1-reproducing; we write $T_\wedge(\psi_1, \psi_2)$ for the formula obtained from $T_{u\wedge v}$ by substituting the formula ψ_1 for u and ψ_2 for v, and similarly write $T^*_\wedge(\psi_1, \psi_2)$ for the formula obtained from $T^*_{u\wedge v}$ in this way. We let $\phi' = \textsc{Tr}(\phi)$, where $\textsc{Tr}$ is the following recursive algorithm that takes a CNF-formula as input:

Algorithm $\textsc{Tr}(\psi_1 \wedge \cdots \wedge \psi_m)$

If $m = 1$, return $T^*_{\psi_1}$.

Else return $T^*_\wedge\Big(\textsc{Tr}(\psi_1 \wedge \cdots \wedge \psi_{\lfloor m/2 \rfloor}), \textsc{Tr}(\psi_{\lfloor m/2 \rfloor + 1} \wedge \cdots \wedge \psi_m)\Big)$.

Since the recursion terminates after a number of steps logarithmic in the number of clauses of ϕ, and every step increases the total formula size by only a constant factor, the algorithm runs in polynomial time. We show $\phi' \equiv T_\phi$ by induction on m. For $m = 1$ this is clear. For the induction step, we have to show $T^*_\wedge(T_{\psi_1}, T_{\psi_2}) \equiv T_{\psi_1\wedge\psi_2}$, but since $T_\wedge(\psi_1, \psi_2) \equiv T^*_\wedge(\psi_1, \psi_2)$, it suffices to show that $T_\wedge(T_{\psi_1}, T_{\psi_2}) \equiv T_{\psi_1\wedge\psi_2}$:

$$T_\wedge(T_{\psi_1}, T_{\psi_2}) = (\psi_1 \wedge y) \wedge (\psi_2 \wedge y) \wedge y \equiv \psi_1 \wedge \psi_2 \wedge y = T_{\psi_1\wedge\psi_2}.$$

2. This follows from 1. by Proposition 4.3.1.

3. By Lemma 2.3.2, there is an 1-reproducing $(n-1)$-ary function f with diameter of at least $2^{\lfloor \frac{n-1}{2} \rfloor}$. Let f be represented by a formula ϕ; then, T_ϕ represents an n-ary function of the same diameter in S_{12}. □

Lemma 4.4.5 *If* $[B] \supseteq \mathsf{D}_1$,

1. $\textsc{st-BF-Conn}(B)$ *and* $\textsc{BF-Conn}(B)$ *are* PSPACE-*complete,*
2. $\textsc{st-Circ-Conn}(B)$ *and* $\textsc{Circ-Conn}(B)$ *are* PSPACE-*complete,*
3. *for* $n \geq 5$, *there is an* n-*ary function* $f \in [B]$ *with diameter of at least* $2^{\lfloor \frac{n-3}{2} \rfloor}$.

Proof. 1. As noted, we adapt the two steps from the previous proof.

Step 1. Since $\mathsf{D}_1 = \mathsf{D} \cap \mathsf{R}_0 \cap \mathsf{R}_1$, T_ψ must be self-dual, 0-reproducing, and 1-reproducing. For clarity, we first construct an intermediate formula $T^\sim_\psi \in \mathsf{D}_1$ whose solution graph has an additional component, then we eliminate that component.

For $\psi(\boldsymbol{x})$, let

$$T^\sim_\psi = (\psi(\boldsymbol{x}) \wedge (\boldsymbol{y} = \boldsymbol{1})) \vee \Big(\overline{\psi(\overline{\boldsymbol{x}})} \wedge (\boldsymbol{y} = \boldsymbol{0})\Big) \vee (\boldsymbol{y} \in \{100, 010, 001\}),$$

where $\boldsymbol{y} = (y_1, y_2, y_3)$ are three new variables.

$T_\psi^\sim$ is self-dual: for any solution ending with 111 (satisfying the first disjunct), the inverse vector is no solution; similarly, for any solution ending with 000 (satisfying the second disjunct), the inverse vector is no solution; finally, all vectors ending with 100, 010, or 001 are solutions and their inverses are no solutions. Also, $T_\psi^\sim$ is still 1-reproducing, and it is 0-reproducing (for the second disjunct note that $\overline{\psi(\overline{0\cdots0})} \equiv \overline{\psi(1\cdots1)} \equiv 0$).

Further, every solution $\boldsymbol{a}$ of ψ corresponds to a solution $\boldsymbol{a}\cdot 111$ of $T_\psi^\sim$, and for any two solutions $\boldsymbol{s}$ and $\boldsymbol{t}$ of ψ, $\boldsymbol{s}' = \boldsymbol{s}\cdot 111$ and $\boldsymbol{t}' = \boldsymbol{t}\cdot 111$ are connected in $G(T_\psi^\sim)$ iff $\boldsymbol{s}$ and $\boldsymbol{t}$ are connected in $G(\psi)$: The "if" is clear, for the "only if" note that since there are no solutions of $T_\psi^\sim$ ending with 110, 101, or 011, every solution of $T_\psi^\sim$ not ending with 111 differs in at least two variables from the solutions that do.

Observe that exactly one connected component is added in $G(T_\psi^\sim)$ to the components corresponding to those of $G(\psi)$: It consists of all solutions ending with 000, 100, 010, or 001 (any two vectors ending with 000 are connected e.g. via those ending with 100). It follows that $G(T_\psi^\sim)$ is always unconnected. To fix this, we modify $T_\psi^\sim$ to T_ψ by adding $1\cdots1\cdot110$ as a solution, thereby connecting $1\cdots1\cdot111$ (which is always a solution since $T_\psi^\sim$ is 1-reproducing) with $1\cdots1\cdot100$, and thereby with the additional component of T_ψ. To keep the function self-dual, we must in turn remove $0\cdots0\cdot001$, which does not alter the connectivity. Formally,

$$\begin{aligned} T_\psi &= \Big(T_\psi^\sim \vee ((\boldsymbol{x}=\mathbf{1}) \wedge (\boldsymbol{y}=110))\Big) \wedge \neg\,((\boldsymbol{x}=\mathbf{0}) \wedge (\boldsymbol{y}=001)) \qquad (4.4.1)\\ &= (\psi(\boldsymbol{x}) \wedge (\boldsymbol{y}=\mathbf{1})) \vee \Big(\overline{\psi(\overline{\boldsymbol{x}})} \wedge (\boldsymbol{y}=\mathbf{0})\Big)\\ &\quad \vee (\boldsymbol{y} \in \{100,010,001\} \wedge \neg((\boldsymbol{x}=\mathbf{0}) \wedge (\boldsymbol{y}=001)))\\ &\quad \vee((\boldsymbol{x}=\mathbf{1}) \wedge (\boldsymbol{y}=110)). \end{aligned}$$

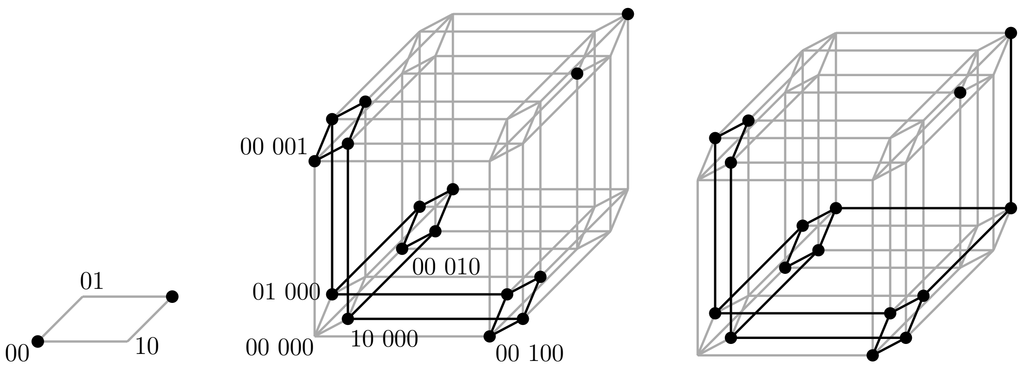


Figure 4.4.1 An example for the transformation. Left: $\psi = (x_1 \vee \overline{x_2}) \wedge (\overline{x_1} \vee x_2)$, center: $T_\psi^\sim$, right: T_ψ. The "axis vertices" are labeled in the first two graphs.

Now $G(T_\psi)$ is connected iff $G(\psi)$ is connected.

Step 2. Again, we use the algorithm Tr from the previous proof to transform any 1-reproducing 3-CNF-formula ϕ into a B-formula ϕ' equivalent to T_ϕ, but with the definition (4.4.1) of T. Again, we have to show $T_\wedge(T_{\psi_1}, T_{\psi_2}) \equiv T_{\psi_1 \wedge \psi_2}$. Here,

$$\begin{aligned} T_\wedge(T_{\psi_1}, T_{\psi_2}) &= (T_{\psi_1} \wedge T_{\psi_2} \wedge (\boldsymbol{y} = \mathbf{1})) \vee \left(\overline{\overline{T_{\psi_1}} \wedge \overline{T_{\psi_2}}} \wedge (\boldsymbol{y} = \mathbf{0})\right) \\ &\quad \vee \left(\boldsymbol{y} \in \{100, 010, 001\} \wedge \neg \left(\overline{T_{\psi_1}} \wedge \overline{T_{\psi_2}} \wedge (\boldsymbol{y} = 001)\right)\right) \\ &\quad \vee (T_{\psi_1} \wedge T_{\psi_2} \wedge (\boldsymbol{y} = 110)) \,. \end{aligned}$$

We consider the parts of the formula in turn: For any formula ξ we have $T_\xi(\boldsymbol{x}_\xi) \wedge (\boldsymbol{y} = \mathbf{1}) \equiv \xi(\boldsymbol{x}_\xi) \wedge (\boldsymbol{y} = \mathbf{1})$ and $T_\xi(\boldsymbol{x}_\xi) \wedge (\boldsymbol{y} = \mathbf{0}) \equiv \overline{\psi(\overline{\boldsymbol{x}_\xi})} \wedge (\boldsymbol{y} = \mathbf{0})$, where $\boldsymbol{x}_\xi$ denotes the variables of ξ. Using $\overline{\overline{T_{\psi_1}(\boldsymbol{x}_{\psi_1})} \wedge \overline{T_{\psi_2}(\boldsymbol{x}_{\psi_2})}} \wedge (\boldsymbol{y} = \mathbf{0}) = (T_{\psi_1}(\boldsymbol{x}_{\psi_1}) \vee T_{\psi_2}(\boldsymbol{x}_{\psi_2})) \wedge (\boldsymbol{y} = \mathbf{0})$, the first line becomes

$$(\psi_1(\boldsymbol{x}_{\psi_1}) \wedge \psi_2(\boldsymbol{x}_{\psi_2}) \wedge (\boldsymbol{y} = \mathbf{1})) \vee \left(\left(\overline{\psi_1(\overline{\boldsymbol{x}_{\psi_1}}) \wedge \psi_2(\overline{\boldsymbol{x}_{\psi_2}})}\right) \wedge (\boldsymbol{y} = \mathbf{0})\right).$$

For the second line, we observe

$$\begin{aligned} \overline{T_\psi(\boldsymbol{x}_\psi)} &\equiv \left(\overline{\psi(\boldsymbol{x}_\psi)} \vee \neg(\boldsymbol{y} = \mathbf{1})\right) \wedge (\psi(\overline{\boldsymbol{x}}_\psi) \vee \neg(\boldsymbol{y} = \mathbf{0})) \\ &\quad \wedge (\boldsymbol{y} \notin \{100, 010, 001\} \vee ((\boldsymbol{x}_\psi = \mathbf{0}) \wedge (\boldsymbol{y} = 001))) \\ &\quad \wedge (\neg(\boldsymbol{x}_\psi = \mathbf{1}) \vee \overline{(\boldsymbol{y} = 110)}), \end{aligned}$$

thus $\overline{T_\psi(\boldsymbol{x}_\psi)} \wedge (\boldsymbol{y} = 001) \equiv (\boldsymbol{x}_\psi = \mathbf{0}) \wedge (\boldsymbol{y} = 001)$, and the second line becomes

$$\vee (\boldsymbol{y} \in \{100, 010, 001\} \wedge \neg ((\boldsymbol{x}_{\psi_1} = \mathbf{0}) \wedge (\boldsymbol{x}_{\psi_2} = \mathbf{0}) \wedge (\boldsymbol{y} = 001))) \,.$$

Since $T_\psi(\boldsymbol{x}_\psi) \wedge (\boldsymbol{y} = 110) \equiv (\boldsymbol{x}_\psi = \mathbf{1}) \wedge (\boldsymbol{y} = 110)$ for any ψ, the third line becomes

$$\vee ((\boldsymbol{x}_{\psi_1} = \mathbf{1}) \wedge (\boldsymbol{x}_{\psi_2} = \mathbf{1}) \wedge (\boldsymbol{y} = 110)) \,.$$

Now $T_\wedge(T_{\psi_1}, T_{\psi_2})$ equals

$$\begin{aligned} T_{\psi_1 \wedge \psi_2} &= (\psi_1(\boldsymbol{x}_{\psi_1}) \wedge \psi_2(\boldsymbol{x}_{\psi_2}) \wedge (\boldsymbol{y} = \mathbf{1})) \vee \left(\overline{\psi_1(\overline{\boldsymbol{x}_{\psi_1}}) \wedge \psi_2(\overline{\boldsymbol{x}_{\psi_2}})} \wedge (\boldsymbol{y} = \mathbf{0})\right) \\ &\quad \vee (\boldsymbol{y} \in \{100, 010, 001\} \wedge \neg ((\boldsymbol{x}_{\psi_1} = \mathbf{0}) \wedge (\boldsymbol{x}_{\psi_2} = \mathbf{0}) \wedge (\boldsymbol{y} = 001))) \\ &\quad \vee ((\boldsymbol{x}_{\psi_1} = \mathbf{1}) \wedge (\boldsymbol{x}_{\psi_2} = \mathbf{1}) \wedge (\boldsymbol{y} = 110)) \,. \end{aligned}$$

2. This follows from 1. by Proposition 4.3.1.

3. By Lemma 2.3.2 there is an 1-reproducing $(n-3)$-ary function f with diameter of at least $2^{\lfloor \frac{n-3}{2} \rfloor}$. Let f be represented by a formula ϕ; then, T_ϕ represents an n-ary function of the same diameter in D_1. □

Lemma 4.4.6 *If* $[B] \supseteq \mathsf{S}_{02}^k$ *for any* $k \geq 2$*,*

1. st-BF-Conn(B) *and* BF-Conn(B) *are* PSPACE*-complete,*
2. st-Circ-Conn(B) *and* Circ-Conn(B) *are* PSPACE*-complete,*
3. *for* $n \geq k+4$*, there is an* n*-ary function* $f \in [B]$ *with diameter of at least* $2^{\lfloor \frac{n-k-2}{2} \rfloor}$.

Proof. 1. *Step 1.* Since $\mathsf{S}_{02}^k = \mathsf{S}_0^k \cap \mathsf{R}_0 \cap \mathsf{R}_1$, T_ψ must be 0-separating of degree k, 0-reproducing, and 1-reproducing. As in the previous proof, we construct an intermediate formula $T_\psi^\sim$. For $\psi(\boldsymbol{x})$, let

$$T_\psi^\sim = (\psi \wedge y \wedge (\boldsymbol{z} = \boldsymbol{0})) \vee (|\boldsymbol{z}| > 1),$$

where y and $\boldsymbol{z} = (z_1, \dots, z_{k+1})$ are new variables.

$T_\psi^\sim(\boldsymbol{x}, y, \boldsymbol{z})$ is 0-separating of degree k, since all vectors that are no solutions of $T_\psi^\sim$ have $|\boldsymbol{z}| \leq 1$, i.e. $\boldsymbol{z} \in \{0\cdots 0, 10\cdots 0, 010\cdots 0, \dots, 0\cdots 01\} \subset \{0,1\}^{k+1}$, and thus any k of them have at least one common variable assigned 0. Also, $T_\psi^\sim$ is 0-reproducing and still 1-reproducing.

Further, for any two solutions $\boldsymbol{s}$ and $\boldsymbol{t}$ of $\psi(\boldsymbol{x})$, $\boldsymbol{s}' = \boldsymbol{s} \cdot 1 \cdot 0 \cdots 0$ and $\boldsymbol{t}' = \boldsymbol{t} \cdot 1 \cdot 0 \cdots 0$ are solutions of $T_\psi^\sim(\boldsymbol{x}, y, \boldsymbol{z})$ and are connected in $G(T_\psi^\sim)$ iff $\boldsymbol{s}$ and $\boldsymbol{t}$ are connected in $G(\psi)$.

But again, we have produced an additional connected component (consisting of all solutions with $|\boldsymbol{z}| > 1$). To connect it to a component corresponding to one of ψ, we add $1 \cdots 1 \cdot 1 \cdot 10 \cdots 0$ as a solution,

$$T_\psi \quad = \quad (\psi \wedge y \wedge (\boldsymbol{z} = \boldsymbol{0})) \vee (|\boldsymbol{z}| > 1) \vee ((\boldsymbol{x} = \boldsymbol{1}) \wedge y \wedge (\boldsymbol{z} = 10 \cdots 0))\,.$$

Now $G(T_\psi)$ is connected iff $G(\psi)$ is connected.

Step 2. Again we show that the algorithm TR works in this case. Here,

$$\begin{aligned} T_\wedge(T_{\psi_1}, T_{\psi_2}) \quad &= \quad (T_{\psi_1}(\boldsymbol{x}_{\psi_1}) \wedge T_{\psi_2}(\boldsymbol{x}_{\psi_2}) \wedge y \wedge (\boldsymbol{z} = \boldsymbol{0})) \vee (|\boldsymbol{z}| > 1) \\ &\quad\ \vee (T_{\psi_1}(\boldsymbol{x}_{\psi_1}) \wedge T_{\psi_2}(\boldsymbol{x}_{\psi_2}) \wedge y \wedge (\boldsymbol{z} = 10 \cdots 0))\,. \end{aligned}$$

Since $T_\psi(\boldsymbol{x}_\psi) \wedge y \wedge (\boldsymbol{z} = \boldsymbol{0}) \equiv \psi(\boldsymbol{x}_\psi) \wedge y \wedge (\boldsymbol{z} = \boldsymbol{0})$ and $T_\psi(\boldsymbol{x}_\psi) \wedge y \wedge (\boldsymbol{z} = 10 \cdots 0) \equiv (\boldsymbol{x}_\psi = 1) \wedge y \wedge (\boldsymbol{z} = 10 \cdots 0)$ for any ψ, this is equivalent to

$$\begin{aligned} T_{\psi_1 \wedge \psi_2} \quad &= \quad (\psi_1(\boldsymbol{x}_{\psi_1}) \wedge \psi_2(\boldsymbol{x}_{\psi_2}) \wedge y \wedge (\boldsymbol{z} = \boldsymbol{0})) \vee (|\boldsymbol{z}| > 1) \\ &\quad\ \vee (\boldsymbol{x}_{\psi_1} \wedge \boldsymbol{x}_{\psi_2} \wedge y \wedge (\boldsymbol{z} = 10 \cdots 0))\,. \end{aligned}$$

2. This follows from 1. by Proposition 4.3.1.

3. By Lemma 2.3.2 there is an 1-reproducing $(n-k-2)$-ary function f with diameter of at least $2^{\lfloor \frac{n-k-2}{2} \rfloor}$. Let f be represented by a formula ϕ; then, T_ϕ represents an n-ary function of the same diameter in S_{02}^k. $\square$

This completes the proof of Theorem 4.2.1.

4.5 Quantified Formulas

Definition 4.5.1 A *quantified B-formula* ϕ (in prenex normal form) is an expression of the form

$$Q_1 y_1 \cdots Q_m y_m \varphi(y_1, \dots, y_m, x_1, \dots, x_n),$$

where φ is a B-formula, and $Q_1, \dots, Q_m \in \{\exists, \forall\}$ are quantifiers. We denote the corresponding connectivity resp. *st*-connectivity problems by QBF-CONN(B) resp. ST-QBF-CONN(B).

Theorem 4.5.2 *Let B be a finite set of Boolean functions.*

1. *If $B \subseteq \mathsf{M}$ or $B \subseteq \mathsf{L}$, then*

   a) st-QBF-Conn(B) *and* QBF-Conn(B) *are in* P*,*

   b) *the diameter of every quantified B-formula is linear in the number of free variables.*

2. *Otherwise,*

   a) st-QBF-Conn(B) *and* QBF-Conn(B) *are* PSPACE*-complete,*

   b) *there are quantified B-formulas with at most one quantifier such that their diameter is exponential in the number of free variables.*

Proof. 1. For $B \subseteq \mathsf{M}$, any quantified B-formula ϕ represents a monotone function: Using $\exists y \psi(y, \boldsymbol{x}) = \psi(0, \boldsymbol{x}) \vee \psi(1, \boldsymbol{x})$ and $\forall y \psi(y, \boldsymbol{x}) = \psi(0, \boldsymbol{x}) \wedge \psi(1, \boldsymbol{x})$ recursively, we can transform ϕ into an equivalent M-formula since $\wedge$ and $\vee$ are monotone. Thus as in Lemma 4.3.2, st-QBF-Conn(B) and QBF-Conn(B) are trivial, and $d_f(\boldsymbol{a}, \boldsymbol{b}) = |\boldsymbol{a} - \boldsymbol{b}|$ for any two solutions $\boldsymbol{a}$ and $\boldsymbol{b}$.

For a quantified B-formula $\phi = Q_1 y_1 \cdots Q_m y_m \varphi$ with $B \subseteq \mathsf{L}$, we first remove the quantifications over all fictive variables of φ (and eliminate the fictive variables if necessary). If quantifiers remain, ϕ is either tautological (if the rightmost quantifier is $\exists$) or unsatisfiable (if the rightmost quantifier is $\forall$), so the problems are trivial, and $d_f(\boldsymbol{a}, \boldsymbol{b}) = |\boldsymbol{a} - \boldsymbol{b}|$ for any two solutions $\boldsymbol{a}$ and $\boldsymbol{b}$. Otherwise, we have a quantifier-free formula and the statements follow from Lemma 4.3.4.

2. Again as in Lemma 2.3.1, it follows that st-QBF-Conn(B) and QBF-Conn(B) are in PSPACE, since the evaluation problem for quantified B-formulas is in PSPACE [Sch78].

An inspection of Post's lattice shows that if $B \not\subseteq \mathsf{M}$ and $B \not\subseteq \mathsf{L}$, then $[B] \supseteq \mathsf{S}_{12}$, $[B] \supseteq \mathsf{D}_1$, or $[B] \supseteq \mathsf{S}_{02}$, so we have to prove PSPACE-completeness and show the existence of B-formulas with an exponential diameter in these cases.

For $[B] \supseteq \mathsf{S}_{12}$ and $[B] \supseteq \mathsf{D}_1$, the statements for the PSPACE-hardness and the diameter obviously carry over from Theorem 4.2.1.

For $B \supseteq \mathsf{S}_{02}$, we give a reduction from the problems for (unquantified) 3-CNF-formulas; we proceeded again similar as in the proof of Lemma 4.4.4. We give a transformation T_ψ s.t. $T_\psi \in \mathsf{S}_{02}$ for all formulas ψ. Since $\mathsf{S}_{02} = \mathsf{S}_0 \cap \mathsf{R}_0 \cap \mathsf{R}_1$, T_ψ must be self-dual, 0-reproducing, and 1-reproducing. For $\psi(\boldsymbol{x})$ let

$$T_\psi = (\psi \wedge y) \vee z,$$

with the two new variables y and z.

T_ψ is 0-separating since all vectors that are no solutions have $z = 0$. Also, T_ψ is 0-reproducing and 1-reproducing. Again, we use the algorithm Tr from the proof of Lemma 4.4.4 to transform any 3-CNF-formula ϕ into a B-formula φ' equivalent to T_ϕ. Again, we show

$$\begin{aligned} T_\wedge(T_{\psi_1}, T_{\psi_2}) &= (((\psi_1 \wedge y) \vee z) \wedge ((\psi_2 \wedge y) \vee z) \wedge y) \vee z \\ &\equiv ((\psi_1 \wedge y) \wedge (\psi_2 \wedge y) \wedge y) \vee z \\ &\equiv (\psi_1 \wedge \psi_2 \wedge y) \vee z = T_{\psi_1 \wedge \psi_2}. \end{aligned}$$

Now let

$$\phi' = \forall z \varphi'.$$

Then, for any two solutions $\boldsymbol{s}$ and $\boldsymbol{t}$ of $\phi(\boldsymbol{x})$, $\boldsymbol{s}' = \boldsymbol{s} \cdot 1$ and $\boldsymbol{t}' = \boldsymbol{t} \cdot 1$ are solutions of $\phi'(\boldsymbol{x}, y)$, and they are connected in $G(\phi')$ iff $\boldsymbol{s}$ and $\boldsymbol{t}$ are connected in $G(\phi)$, and $G(\phi')$ is connected iff $G(\phi)$ is connected.

The proof of Lemma 2.3.2 shows that there is an $(n-1)$-ary function f with diameter of at least $2^{\lfloor \frac{n-1}{2} \rfloor}$. Let f be represented by a formula ϕ; then ϕ' as defined above is a quantified B-formula with n free variables and one quantifier with the same diameter. $\square$

Remark 4.5.3. An analog to Theorem 4.5.2 also holds for quantified circuits as defined in [RW00, Section 7].

5 Future Directions

We have proved classifications in two quite different settings: In Schaefer's framework for constraint satisfaction problems, specifically $\mathrm{CNF_C}(\mathcal{S})$-formulas, $\mathrm{CNF}(\mathcal{S})$-formulas and Q-$\mathrm{CNF_C}(\mathcal{S})$-formulas, and in Post's framework for nested formulas and circuits, specifically B-formulas, B-circuits, and quantified B-formulas.

While we now have a quite complete picture for nested formulas and circuits, and also for $\mathrm{CNF_C}(\mathcal{S})$-formulas and Q-$\mathrm{CNF_C}(\mathcal{S})$-formulas, the complexity of the connectivity problem for $\mathrm{CNF}(\mathcal{S})$-formulas without constants is still open for sets $\mathcal{S}$ that are 0-valid, 1-valid, or complementive, but not Schaefer, nor nc-CPSS, nor quasi disconnecting.

Recently, Scharpfenecker refined some of our complexity results for $\mathrm{CNF_C}(\mathcal{S})$-formulas up to logarithmic-space isomorphisms and investigated the realizable solution-graphs in more detail [Sch15].

It seems very likely that connectivity is not for all these sets in P, since there are even Schaefer sets that are 0-valid or 1-valid, but have a coNP-complete connectivity problem. On the other hand, it seems promising to search for more nc-CPS sets, for which we have a P or $\mathrm{P^{NP}}$ algorithm.

Also, one might further explore the connection between the connectivity of CNF-formulas and properties of the corresponding hypergraphs (see Remark 2.7.3); while we used this connection only for Horn formulas and thus dealt with hypergraphs of head-size 1, there might be useful reductions between the connectivity problems for more general CNF-formulas and problems for the corresponding hypergraphs.

There is a multitude of interesting variations of the problems investigated in this thesis, in different directions; we close with a quick survey of some such variations.

- **Other representations of Boolean relations** Disjunctive normal forms with special connectivity properties were studied by Ekin et al. already in 1997 for their "important role in problems appearing in various areas including in particular discrete optimization, machine learning, automated reasoning, etc." [EHK99].

  There are yet more kinds of representations for Boolean relations, such as binary decision diagrams and Boolean neural networks, and investigating the connectivity in these settings might be worthwhile as well.

- **Related problems** Other connectivity-related problems already mentioned by Gopalan et al. are counting the number of components and approximating the diameter. For counting the number of components, Gopalan et al. mentioned in [GKMP06] that they could show that the problem is in P for affine, monotone and dual-monotone relations, and #P-complete otherwise.

  Further, especially with regard to reconfiguration problems, it is interesting to find the shortest path between two solutions; this was recently investigated by Mouawad et al. [MNPR14], who proved a computational trichotomy for this problem. In this direction, one could also consider the optimal path according to some other measure.

- **Other definitions of connectivity** Our definition of connectivity is not the only sensible one: One could regard two solutions connected whenever their Hamming distance is at most d, for some $d \geq 1$; this was already considered related to random satisfiability, see [ART06]. This generalization seems meaningful as well as challenging.

- **Higher domains** Finally, a most interesting subject are CSPs over larger domains; in 1993, Feder and Vardi conjectured a dichotomy for the satisfiability problem over arbitrary finite domains [FV98], and while the conjecture was proved for domains of size three in 2002 by Bulatov [Bul02], it remains open to date for the general case. Close investigation of the solution space might lead to valuable insights here.

  For k-colorability, which is a special case of the general CSP over a k-element set, the connectivity problems and the diameter were already studied by Bonsma and Cereceda [BC09], and Cereceda, van den Heuvel, and Johnson [CvdHJ11]. They showed that for $k = 3$ the diameter is at most quadratic in the number of vertices and the st-connectivity problem is in P, while for $k \geq 4$, the diameter can be exponential and st-connectivity is PSPACE-complete in general.

Bibliography

[AP01] Kim Allan Andersen and Daniele Pretolani, *Easy cases of probabilistic satisfiability*, Annals of Mathematics and Artificial Intelligence **33** (2001), no. 1, 69–91. 25

[APT79] Bengt Aspvall, Michael F. Plass, and Robert Endre Tarjan, *A linear-time algorithm for testing the truth of certain quantified boolean formulas*, Inf. Process. Lett. **8** (1979), no. 3, 121–123. 54

[ART06] Dimitris Achlioptas and Federico Ricci-Tersenghi, *On the solution-space geometry of random constraint satisfaction problems*, Proceedings of the thirty-eighth annual ACM symposium on Theory of computing, ACM, 2006, pp. 130–139. 70

[Bau07] Michael Bauland, *Complexity results for boolean constraint satisfaction problems*, Ph.D. thesis, 2007. 50, 54

[BB08] Uwe Bubeck and Hans Kleine Büning, *Models and quantifier elimination for quantified horn formulas*, Discrete Applied Mathematics **156** (2008), no. 10, 1606–1622. 52

[BC09] Paul Bonsma and Luis Cereceda, *Finding paths between graph colourings: Pspace-completeness and superpolynomial distances*, Theoretical Computer Science **410** (2009), no. 50, 5215–5226. 70

[BCRV03] Elmar Böhler, Nadia Creignou, Steffen Reith, and Heribert Vollmer, *Playing with boolean blocks, part i: Posts lattice with applications to complexity theory*, SIGACT News, 2003. 9, 55, 56

[BRSV05] Elmar Böhler, Steffen Reith, Henning Schnoor, and Heribert Vollmer, *Bases for boolean co-clones*, Information Processing Letters **96** (2005), no. 2, 59–66. 9

[Bul02] Andrei A Bulatov, *A dichotomy theorem for constraints on a three-element set*, Foundations of Computer Science, 2002. Proceedings. The 43rd Annual IEEE Symposium on, IEEE, 2002, pp. 649–658. 70

[CH96] Nadia Creignou and Miki Hermann, *Complexity of generalized satisfiability counting problems*, Information and Computation **125** (1996), no. 1, 1–12. 47

[CKS01] Nadia Creignou, Sanjeev Khanna, and Madhu Sudan, *Complexity classifications of boolean constraint satisfaction problems*, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2001. 8, 9

[CKV08] Nadia Creignou, Phokion G. Kolaitis, and Heribert Vollmer, *Complexity of constraints*, 2008. 12

[CKZ08] Nadia Creignou, Phokion Kolaitis, and Bruno Zanuttini, *Structure identification of boolean relations and plain bases for co-clones*, Journal of Computer and System Sciences **74** (2008), no. 7, 1103–1115. 2

[CvdHJ11] Luis Cereceda, Jan van den Heuvel, and Matthew Johnson, *Finding paths between 3-colorings*, Journal of graph theory **67** (2011), no. 1, 69–82. 70

[EHK99] Oya Ekin, Peter L Hammer, and Alexander Kogan, *On connected boolean functions*, Discrete Applied Mathematics **96** (1999), 337–362. 69

[FM07] Zhaohui Fu and Sharad Malik, *Extracting logic circuit structure from conjunctive normal form descriptions*, VLSI Design, 2007. Held jointly with 6th International Conference on Embedded Systems., 20th International Conference on, IEEE, 2007, pp. 37–42. 3

[FV98] Tomás Feder and Moshe Y Vardi, *The computational structure of monotone monadic snp and constraint satisfaction: A study through datalog and group theory*, SIAM Journal on Computing **28** (1998), no. 1, 57–104. 70

[GKMP06] Parikshit Gopalan, Phokion G. Kolaitis, Elitza N. Maneva, and Christos H. Papadimitriou, *The connectivity of boolean satisfiability: Computational and structural dichotomies*, ICALP'06, 2006, pp. 346–357. 3, 35, 69

[GKMP09] Parikshit Gopalan, Phokion G. Kolaitis, Elitza Maneva, and Christos H. Papadimitriou, *The connectivity of boolean satisfiability: Computational and structural dichotomies*, SIAM J. Comput. **38** (2009), no. 6, 2330–2355. 3, 9, 10, 12, 13, 14, 19, 20, 21, 22, 24, 26, 52, 54, 60, 61

[GLPN93] Giorgio Gallo, Giustino Longo, Stefano Pallottino, and Sang Nguyen, *Directed hypergraphs and applications*, Discrete applied mathematics **42** (1993), no. 2, 177–201. 25

[HD02] Robert A Hearn and Erik D Demaine, *The nondeterministic constraint logic model of computation: Reductions and applications*, Automata, Languages and Programming, Springer, 2002, pp. 401–413. 19

[IDH+11] Takehiro Ito, Erik D. Demaine, Nicholas J. A. Harvey, Christos H. Papadimitriou, Martha Sideri, Ryuhei Uehara, and Yushi Uno, *On the complexity of reconfiguration problems*, Theor. Comput. Sci. **412** (2011), no. 12-14, 1054–1065. 2

[KMM11] Marcin Kamiński, Paul Medvedev, and Martin Milanič, *Shortest paths between shortest paths and independent sets*, Combinatorial Algorithms, Springer, 2011, pp. 56–67. 2

[Lew79] Harry R Lewis, *Satisfiability problems for propositional calculi*, Mathematical Systems Theory **13** (1979), no. 1, 45–53. 56

[MMW07] Elitza Maneva, Elchanan Mossel, and Martin J Wainwright, *A new look at survey propagation and its generalizations*, Journal of the ACM (JACM) **54** (2007), no. 4, 17. 3

[MMZ05] Marc Mézard, Thierry Mora, and Riccardo Zecchina, *Clustering of solutions in the random satisfiability problem*, Physical Review Letters **94** (2005), no. 19, 197205. 3

[MNPR14] Amer E Mouawad, Naomi Nishimura, Vinayak Pathak, and Venkatesh Raman, *Shortest reconfiguration paths in the solution space of boolean formulas*, arXiv preprint (2014). 69

[MTY07] Kazuhisa Makino, Suguru Tamaki, and Masaki Yamamoto, *On the boolean connectivity problem for horn relations*, Proceedings of the 10th international conference on Theory and applications of satisfiability testing, SAT'07, 2007, pp. 187–200. 3, 24, 54

[Pos41] Emil L Post, *The two-valued iterative systems of mathematical logic.(am-5)*, vol. 5, Princeton University Press, 1941. 56

[RW99] Steffen Reith and Klaus W. Wagner, *The complexity of problems defined by subclasses of boolean functions*, Tech. report, 1999. 56

[RW00] Steffen Reith and Klaus W Wagner, *The complexity of problems defined by boolean circuits*, 2000. 56, 67

[Sch78] Thomas J. Schaefer, *The complexity of satisfiability problems*, STOC '78, 1978, pp. 216–226. 7, 12, 35, 36, 50, 66

[Sch07] Henning Schnoor, *Algebraic techniques for satisfiability problems*, Ph.D. thesis, Universität Hannover, 2007. 56

[Sch13] Konrad W Schwerdtfeger, *A computational trichotomy for connectivity of boolean satisfiability*, Journal on Satisfiability, Boolean Modeling and Computation **8** (2013), 173–195, Corrected version at `http://arxiv.org/abs/1312.4524`. 3, 8, 10, 14

[Sch14a] ———, *The connectivity of boolean satisfiability: Dichotomies for formulas and circuits*, Proceedings of the 9th International Computer Science Symposium in Russia (CSR 2014) (2014), 351–364, `http://arxiv.org/abs/1312.6679`. 4

[Sch14b] ———, *The connectivity of boolean satisfiability: No-constants and quantified variants*, arXiv CoRR (2014), `http://arxiv.org/abs/1403.6165`. 4

[Sch15] Patrick Scharpfenecker, *On the structure of solution-graphs for boolean formulas*, Fundamentals of Computation Theory, Springer International Publishing, 2015, pp. 118–130. 69

[Tho12] Michael Thomas, *On the applicability of post's lattice*, Information Processing Letters **112** (2012), no. 10, 386–391. 56

[Vol99] Heribert Vollmer, *Introduction to circuit complexity: A uniform approach*, Springer-Verlag New York, Inc., 1999. 55

[WLLH07] Chi-An Wu, Ting-Hao Lin, Chih-Chun Lee, and Chung-Yang Ric Huang, *Qutesat: a robust circuit-based sat solver for complex circuit structure*, Proceedings of the conference on Design, automation and test in Europe, EDA Consortium, 2007, pp. 1313–1318. 3

Lebenslauf

Name	Konrad Widukind Schwerdtfeger

6. 8. 1985	in Hildesheim geboren
1992 – 1996	Grundschule Nordstemmen
1996 – 2005	Gymnasium Andreanum, Hildesheim
2005	Abitur
2005 – 2006	Wehrdienst
2005 – 2010	Studium der Physik an der Leibniz Universität Hannover
2010	Diplom in theoretischer Physik
2010 – 2015	Promotionsstudium am Institut für Theoretische Informatik der Universität Hannover