Generalized Satisfiability Problems

Dissertation zur Erlangung des
naturwissenschaftlichen Doktorgrades
der Bayerischen Julius-Maximilians-Universität Würzburg

vorgelegt von

Steffen Reith

aus

Würzburg

Würzburg, 2001

Eingereicht am: 22. 02. 2001
bei der Fakultät für Mathematik und Informatik

1. Gutachter: Prof. Dr. Klaus W. Wagner
2. Gutachter: Prof. Dr. Nadia Creignou

Tag der mündlichen Prüfung: 16. 07. 2001

Ich möchte allen danken, die mit Rat, Unterstützung und Hilfe dazu beigetragen haben, dass diese Arbeit gedeihen konnte. Insbesondere möchte ich Klaus W. Wagner danken, der mir diese Arbeit ermöglichte. Auch meinen Freunden und Kollegen Herbert Baier, Elmar Böhler, Helmut Celina, Matthias Galota, Sven Kosub und Heribert Vollmer möchte ich für die vielen Diskussionen, Ideen, Tipps, Hilfen und Aufmunterungen danken, ohne die diese Arbeit nie entstanden wäre.

Für Christiane

Table of Contents

List of Figures

1. Overview

In the last 40 years, complexity theory has grown to a rich and powerful field in theoretical computer science. The main task of complexity theory is the classification of problems with respect to their consumption of resources (e.g., running time, required memory or number of needed processors).

To study the computational complexity (i.e., consumption of resources) of problems, all "similar" problems are grouped into a "complexity class". The main intention behind this is the hope to get a better insight into the common properties of these problems and therefore to obtain a better understanding of why some problems are easy to solve, whereas for others practical algorithms are either not known or are even proven not to exist.

Clearly, when we want to measure the consumption of resources which are needed to solve a problem, we have to talk about the computing device we want to use. In the history of complexity theory, the *Turing-machine* in various modifications has proven to be a good, robust, and realistic model for this task. Hence all results in this thesis are based on Turing-machines as computational devices.

Everybody who is familiar with the task of finding an algorithm for a given problem is aware of the following dilemma: Often one is able to find a suitable algorithm very fast, others are trickier to find. In some other cases no fast algorithm is known despite tremendous efforts to find one. This led to the situation that by 1970, many problems with no known fast algorithm had been discovered, and it was unknown, whether fast algorithms either did not exist or whether so far nobody had been clever enough to find one.

A major breakthrough in this situation was the introduction of the complexity classes **P** and **NP**, which were introduced by Stephan A. Cook in 1971 (cf. [Coo71]). (Implicitly Leonid Levin achieved the same results independently in [Lev73].) While **P** is the class of problems which can be solved efficiently, i.e., in polynomial time, the class **NP** consists of all problems which can be solved by a *non deterministic Turing-machine* in polynomial time. Today the class **P** is used as a "definition" of computational "easiness" or "tractability"; we call a problem, which is solvable in polynomial time by a deterministic Turing-machine "easy", whereas all other problems are called "intractable". Hence the classes **P** and **NP** are of immense importance.

The problem whether these two classes coincide or differ is known nowadays as the famous $\mathbf{P} \stackrel{?}{=} \mathbf{NP}$ problem. Nearly everybody who is aware of the basics of algorithms is familiar with this problem, showing that it has a huge influence on all aspects of theoretical computer science and especially on complexity theory. Therefore it is not astonishing, that nearly all results in complexity theory are related to this question and also the present thesis is connected to the $\mathbf{P} \stackrel{?}{=} \mathbf{NP}$ problem.

Interestingly, the $\mathbf{P} \stackrel{?}{=} \mathbf{NP}$ problem has a history at least reaching back to 1956, when Gödel sent a letter to von Neumann, in which he stated the following problem:

> Given a first-order formula[1] F and a natural number n, determine if there is a proof of F of length n.

Gödel was interested in the number of steps needed by a Turing-machine to solve this problem and asked whether there is an algorithm for this problem running in linear or quadratic time[2].

Nowadays, a problem which is closely related to Gödel's problem is known as SAT and is used in varying contexts of complexity theory. Moreover SAT was identified by Cook as an example for a problem which is most likely not solvable by any polynomial time algorithm. To justify this statement, he proved the completeness of SAT for **NP**, which means that SAT is one of the hardest problems in the class **NP**. To classify problems in **NP** we have to compare them with all other problems in **NP**. As a tool for comparing the computational complexity of problems Cook introduced the concept of *reductions*. The intention behind reductions is to partially order problems (out of **NP**) with respect to the computational power needed to solve them. Informally, a reduction compares the complexity of two problems in the following way: a problem A *can be reduced to a problem* B if there is an easy transformation which converts an input x for A into an input x' for B and x belongs to A iff x' belongs to B. This means that we can solve A with the help of B and the easy transformation, hence B cannot be easier than A. If we are able to find a problem inside **NP**, which has the property that it is not easier than any other problem in **NP**, then this problem is one of the hardest in **NP**. In Cook's terminology such a problem is called **NP**-*complete* and has a very special property: If we are able to find a polynomial time algorithm for an **NP**-complete problem, then $\mathbf{P} = \mathbf{NP}$ and if we are able to show that $\mathbf{P} \neq \mathbf{NP}$ then we know that there cannot exist an efficient method for solving such an **NP**-complete problem. Because of this property we can imagine that we have distilled all substantial properties from all problems of **NP** into one problem. This fact makes **NP**-complete problems so important for complexity theory.

Another important way of looking at the $\mathbf{P} \stackrel{?}{=} \mathbf{NP}$ problem comes from proof-theory. It is known that **P** can be considered as the class of all problems A such that, for any input x for A, we can find a *proof* or *witness* for the membership of x efficiently, if $x \in A$ holds. On the other hand, **NP** is the class of all problems

[1] Formel des "engeren Funktionenkalküls".

[2] Such an algorithm would mean "dass die Anzahl der Schritte gegenüber dem blossen Probieren von N auf $\log N$ (oder $(\log N)^2$) verringert werden kann".

B such that there exists a short proof of membership, which can be checked efficiently. Take as an example the problem SAT. The input of this problem is a propositional formula H. If $H \in$ SAT we know that there is an assignment of all variables occurring in H, such that this assignment satisfies H. Since we can efficiently check for an arbitrary assignment whether it satisfies a given formula or not, the existence of a satisfying assignment is a proof of the membership of H to SAT. Since such an assignment is short with respect to the length of the formula H, this shows that SAT belongs to **NP**. In contrast to this use the problem HORN-SAT, where we have to check the existence of a satisfying assignment of a Horn-formula. There exists a well known fast algorithm to determine a satisfying assignment of a Horn-formula iff one exists (see [Pap94]), hence a witness for $H \in$ HORN-SAT can be easily computed, showing that HORN-SAT belongs to the class **P**.

But the concept of **NP**-completeness is also important for the design of algorithms. If we can prove that a given problem is **NP**-complete, we have shown that all known tools for designing algorithms cannot help us to manage it, because nobody was able to show $\mathbf{P} = \mathbf{NP}$ up to now. Moreover, it is likely that we will never find an efficient algorithm, because it is widely believed in complexity theory that $\mathbf{P} \neq \mathbf{NP}$. The importance for practical purposes is supported by a steadily growing list of **NP**-complete problems, which belong to logics, networking, storage and retrieval, graph theory, algebra, and number theory. An early collection of more than 300 **NP**-complete problems can be found in [GJ79].

In complexity theory, the study of **NP** led to numerous related classes, e.g., the classes of the polynomial-time hierarchy **PH**, **PP**, **L**, $\#\mathbf{P}$, and $\#\mathbf{L}$. (For a diagram showing some inclusional relationships see Fig. 2.1.) The concept of completeness is not only restricted to the class **NP**. Less well known as **NP**-complete problems, there are also many **P**-complete problems examined today. Many of these problems are related to Boolean formulas and Boolean circuits, e.g., the problem of finding a satisfying assignment of a Horn-formula, which plays an important role in the theory of databases and programming languages. A source of **P**-complete problems having a similar intention as [GJ79] is [GHR95]. Today we know complete problems for subclasses of **P**, like **NL**, $\oplus\mathbf{L}$ and **L**, too.

As mentioned above, Cook showed that SAT is an **NP**-complete problem and various modifications and restrictions of SAT were used to show completeness results, not only for **NP**, but, e.g., for **P**, **NL**, $\oplus\mathbf{L}$, **PP**, $\mathbf{\Sigma}^{\mathbf{p}}_{\boldsymbol{k}}$, and $\#\mathbf{P}$. Hence some questions arise: Why is SAT so hard to solve? Are there restrictions of SAT which are not so hard to solve? Can we prove that these restricted versions of SAT are complete for well-known subclasses of **NP**? And the other way around: How do we have to generalize the SAT-problem to obtain completeness results beyond **NP**, maybe for the classes of the polynomial time hierarchy, **PP** or **PSPACE**? And finally: Do the results for circuits and formulas differ? The present work will focus on these questions.

Boolean functions

In the 1920s, Emil Leon Post extensively studied Boolean functions. Almost twenty years later he published his work in [Pos41], where he presented his results comprehensively. Today these superb results are not widely known and the only other source for his results is the monograph [JGK70], which is a German translation of a Russian original, giving a summary of [Pos41]. One of his main outcomes was the identification of all closed sets of Boolean functions, showing that all those classes are generated by a finite base. For this, he used the *superposition* of Boolean functions as the operation to combine them, where we are allowed to identify, permute, and substitute variables to construct a new function. Intuitively, this is the same as combining Boolean functions by soldering inputs and outputs together, forming a new Boolean function.

This impressive result is not widely known, but the resulting completeness test for Boolean functions can be found in basic textbooks about theoretical computer science (e.g., [Wag94]). We call a set of Boolean functions *complete* iff its closure under superposition is the set of all Boolean functions. Post showed that there are five maximal closed classes of Boolean function; these are circled bold in Figure 2.2 (these maximal closed classes are known as "*Post's classes*"). Hence we know that a set of Boolean functions, which does not belong to any of these five classes, must be a complete set (cf. Corollary 2.10).

Generalized propositional formulas and circuits

When we define propositional formulas inductively, we start with propositional variables. In the induction step, we are allowed to combine smaller propositional formulas with connector symbols. It is common to use symbols for the Boolean *and*, *or* and *not* function, since it is known that all Boolean functions can be represented in this way. Similarly, this technique is used to introduce Boolean circuits. The only difference between circuits and formulas is that circuits have the structure of an acyclic directed graph, whereas we restrict formulas to a tree-like form. We will see later that this restriction of formulas leads to different results in some cases. Now it is natural to generalize circuits and formulas in the following way: We are allowed to use connection symbols which represent Boolean functions out of a fixed set B of Boolean functions. Later we will use the term B-*circuit* or B-*formula* for these generalized Boolean formulas and circuits. Moreover, we will see that B-circuits and B-formulas represent all Boolean functions out of the closure under superposition of B. Because of this, we can make use of Post's results to study our generalized circuits and formulas. Since all closed sets of Boolean functions are known, we are able to give results about "all definable generalized propositional logics" by varying the used set B. Similarly, we are able to study problems related to circuits built over arbitrary bases, i.e., sets of Boolean functions used as gate-types (cf. Chapter 3 and Chapter 4).

Boolean constraint satisfaction problems

Let us start with a prototypical example for constraint satisfaction problems. 3-SAT is a very well-known restriction of SAT, which is still **NP**-complete. Here we have to search for satisfying assignments of a propositional formula in conjunctive normal form, i.e., all inputs are of the form $C_1 \wedge C_2 \wedge \cdots \wedge C_m$, where each C_i (*clause*) consists of a disjunction of Boolean variables, which might be negated. Hence a satisfying assignment has to fulfill each clause in parallel. Therefore, each clause is called a "*constraint*" for our overall solution. But such clauses are not very general and they are hardly useful to model "natural" constraints. To overcome this restriction, in 1978, Thomas Schaefer invented a strong generalization of this concept. He replaced the clauses with an arbitrary Boolean predicate out of a fixed set S and called the resulting formulas S-*formulas*. Now it is easy to define satisfiability problems, like, e.g., EXACTLY-ONE-IN-THREE-SAT, where we require that a satisfying assignment assigns exactly one variable of each clause to true. While studying this kind of formulas, he gave a broad criterion to decide the complexity of this family of satisfiability problems and was able to show that the complexity only depends on the used set of predicates. Very surprisingly, Schaefer proved that the satisfiability problem for S-formulas is either decideable in **P** or complete for **NP**. Hence he called this theorem a *dichotomy theorem* for generalized constraint satisfaction. This is an interesting, surprising, and not very obvious property, because Ladner showed in [Lad75] that there are infinitely many *degrees* between **P** and **NP**, under the assumption that $\mathbf{P} \neq \mathbf{NP}$, hence there is no evident reason why none of Schaefers problems can be found in one of these intermediate degrees. Influenced by Schaefers results, many other problems related to S-formulas were studied (see [Cre95, CH96, KST97, KSW97, CKS99, KK01] and Section 5.4.), leading to a wealth of completeness results for e.g., **PSPACE**, **#P**, and **MAXSNP**. Following this branch of complexity theory, in Section 5.4 we will study the problem of finding the smallest (largest, resp.) satisfying assignment of an S-formula.

A short summary

In Chapter 2 we will give a short introduction into the topics of complexity theory which will be needed here. For a comprehensive introduction into complexity theory see [Pap94, BDG95, BDG90]. After that we advance to Boolean functions and give a short summary about closed classes of Boolean functions. This section heavily hinges on results achieved by Emil Leon Post already in the twenties of the 20^{th} century (see [Pos41]). We finish this chapter with the introduction of generalized Boolean circuits and formulas and some basic notations for them.

In Chapter 3 we will study various problems related to generalized Boolean circuits and formulas. First we will study the circuit value problem for generalized Boolean circuits in Section 3.2, since it plays a central role for all other problems which will be considered in this work. After that we will completely classify the satisfiability and tautology problem for generalized circuits, which leads various completeness results for **NP**, **coNP** and classes below them (cf. Section 3.3). As a natural generalization we will study quantified generalized Boolean circuits in Section 3.4 and obtain results for quantified circuits having a bounded or unbounded number of quantifiers. Additionally there can be found results about the complexity of determining the number of satisfying assignments of generalized circuits and of checking whether a given circuit has more than k satisfying assignments. At the end of this chapter we present a beautiful result by Harry Lewis about generalized formulas and we use it to study the satisfiability problem related to B-formulas.

In Chapter 4 we will present results for the *equivalence problem* and the *isomorphism problem* of B-circuits and B-formulas. This means that we completely clarify the complexity whether two given circuits or formulas represent the same Boolean function. In the isomorphism case, this is beyond the means of present techniques, hence in some cases we only give lower bounds, which are as good as the trivial lower bounds known for the unrestricted isomorphism problem.

Finally in Chapter 5 we study the complexity of finding the lexicographically smallest and largest satisfying assignment for a given generalized formula. In contrast to the other chapters we study two different kinds of formulas, namely B-formulas and S-formulas. After a brief introduction into relevant known results, we turn to the generalized formulas in the Post context. After that we conclude with formulas initially introduced by Thomas Schaefer in [Sch78].

Finally note that Chapter 3 is based on [RW00] and that Chapter 5 was published in [RV00]. The material which will be presented in Chapter 4 was not published before.

2. Basics

2.1 Complexity theory

In this section we will informally introduce some standard notations and results from complexity theory. For more details see [BDG95, BDG90, Pap94].

The complexity classes **P**, **NP**, **L**, **NL**, and **PSPACE** are defined as the classes of languages (problems) which can be accepted by deterministic polynomially time bounded Turing-machines, nondeterministic polynomially time bounded Turing-machines, deterministic logarithmically space bounded Turing-machines, nondeterministic logarithmically space bounded Turing-machines, and deterministic polynomially space bounded Turing-machines, resp. Let **PP** be the class of languages which can be accepted by a nondeterministic polynomially time bounded Turing-machine M in the following manner: An input (x, k) is accepted by M iff more than k of M's computation paths on input x are accepting paths. Let $\oplus\mathbf{L}$ be the class of languages which can be accepted by a nondeterministic logarithmically space bounded Turing-machine M in the following manner: An input x is accepted by M iff the number of M's accepting paths on input x is odd.

Following the notation from [BDG95, BDG90] we use a generic notation for time- and space-bounded classes of problems, which can be solved by deterministic and nondeterministic multi-tape Turing-machines; $\mathbf{DTIME}(s)$, $\mathbf{DSPACE}(s)$, $\mathbf{NTIME}(s)$ and $\mathbf{NSPACE}(s)$, where $s\colon \mathbb{N} \to \mathbb{N}$ is the given bound for time or space. Now $\mathbf{L} = \mathbf{DSPACE}(\log n)$, $\mathbf{NL} = \mathbf{NSPACE}(\log n)$, $\mathbf{P} = \mathbf{DTIME}(n^{O(1)})$, $\mathbf{NP} = \mathbf{NTIME}(n^{O(1)})$ and $\mathbf{PSPACE} = \mathbf{DSPACE}(n^{O(1)})$.

For a class $\mathcal{K}$ of languages let $\mathbf{co}\mathcal{K} =_{\text{def}} \{\overline{A} \mid A \in \mathcal{K}\}$. Furthermore let $\mathbf{P}^{\mathcal{K}}$, $\mathbf{NP}^{\mathcal{K}}$, $\mathbf{L}^{\mathcal{K}}$, and $\mathbf{NL}^{\mathcal{K}}$ be the classes of languages accepted by the type of machines used for the classes **P**, **NP**, **L**, and **NL**, resp., but which have the additional ability to ask queries to languages from $\mathcal{K}$ free of charge (such machines are called *oracle Turing machines*). If we want to restrict the number of queries to s, we will write $\mathbf{P}^{\mathcal{K}}[s]$, $\mathbf{NP}^{\mathcal{K}}[s]$, $\mathbf{L}^{\mathcal{K}}[s]$, and $\mathbf{NL}^{\mathcal{K}}[s]$.

The classes of the *polynomial time hierarchy* are inductively defined as follows: $\mathbf{\Sigma}_0^{\mathrm{p}} = \mathbf{\Pi}_0^{\mathrm{p}} = \mathbf{\Theta}_0^{\mathrm{p}} = \mathbf{\Delta}_0^{\mathrm{p}} =_{\text{def}} \mathbf{P}$. For $k \geq 1$ we define $\mathbf{\Sigma}_k^{\mathrm{p}} =_{\text{def}} \mathbf{NP}^{\mathbf{\Sigma}_{k-1}^{\mathrm{p}}}$, $\mathbf{\Pi}_k^{\mathrm{p}} =_{\text{def}} \mathbf{co}\mathbf{\Sigma}_k^{\mathrm{p}} = \mathbf{coNP}^{\mathbf{\Sigma}_{k-1}^{\mathrm{p}}}$, $\mathbf{\Theta}_k^{\mathrm{p}} =_{\text{def}} \mathbf{P}^{\mathbf{\Sigma}_{k-1}^{\mathrm{p}}}[\log n]$, $\mathbf{\Delta}_k^{\mathrm{p}} =_{\text{def}} \mathbf{P}^{\mathbf{\Sigma}_{k-1}^{\mathrm{p}}}$ and $\mathbf{PH} =_{\text{def}} \bigcup_{k \geq 0} \mathbf{\Sigma}_k^{\mathrm{p}}$.

Let **FL** (**FP**, resp.) be the class of functions which can be computed by a deterministic logarithmically space (deterministic polynomially time, resp.) bounded

Turing-machine. For a nondeterministic polynomially time bounded machine M let $\#M(x)$ be the number of accepting paths of M on input x. Let $\#\mathbf{P}$ be the class of all such functions $\#M$. For a class $\mathcal{K}$ of sets and a function $s\colon \mathbb{N} \to \mathbb{N}$, let $\mathbf{FL}_{\|}^{\mathcal{K}}[s]$ be the class of all functions which are computable by a logspace bounded Turing-machine which, for inputs of length n, can make at most $s(n)$ parallel queries (i.e., the list of all queries is formed before any of them is asked) to a language from $\mathcal{K}$.

Here are some basic facts on the complexity classes defined above. A graphical overview is given in Figure 2.1.

Theorem 2.1 ([Sav70]). *Let $s(n) \geq \log n$ be a space constructible function. Then the following inclusion holds:*

$$\mathbf{NSPACE}(s) \subseteq \mathbf{DSPACE}(s^2).$$

A proof for Theorem 2.1 can also be found in [BDG95], Corollary 2.28.

Theorem 2.2 ([Sze87, Imm88]). *Let $s(n) \geq \log n$ be a space constructible function. Then the following equality holds:*

$$\mathbf{NSPACE}(s) = \mathbf{coNSPACE}(s).$$

A proof for Theorem 2.2 can also be found in [BDG95], Theorem 2.26.

Theorem 2.3. *1.* $\mathbf{L} \subseteq \mathbf{NL} \cap \oplus\mathbf{L} \subseteq \mathbf{NL} \cup \oplus\mathbf{L} \subseteq \mathbf{P} \subseteq \mathbf{NP} \subseteq \mathbf{PSPACE}$.
2. $\mathbf{NL} \cup \oplus\mathbf{L} \subseteq \mathbf{DSPACE}(\log^2 n)$.
3. *The classes* $\mathbf{L}$, $\mathbf{NL}$, $\oplus\mathbf{L}$, $\mathbf{P}$, $\mathbf{PP}$, *and* $\mathbf{PSPACE}$ *are closed under complementation.*
4. $\mathbf{\Sigma}_{\boldsymbol{k}}^{\mathbf{p}} \cup \mathbf{\Pi}_{\boldsymbol{k}}^{\mathbf{p}} \subseteq \mathbf{\Sigma}_{\boldsymbol{k+1}}^{\mathbf{p}} \cap \mathbf{\Pi}_{\boldsymbol{k+1}}^{\mathbf{p}}$ *for* $k \geq 1$ *and* $\bigcup_{k\geq 1}(\mathbf{\Sigma}_{\boldsymbol{k}}^{\mathbf{p}} \cup \mathbf{\Pi}_{\boldsymbol{k}}^{\mathbf{p}}) \subseteq \mathbf{PSPACE}$.
5. $\mathbf{FP} \subseteq \#\mathbf{P}$.
6. *For every $f \in \#\mathbf{P}$ there exist $f' \in \#\mathbf{P}$ and a logspace computable function g such that $f(x) = g(x) - f'(x)$ for all x.*

Proof. The first inclusion of Statement 1 is obvious. For the other inclusions see [BDG95], Theorem 2.26.

The inclusion $\mathbf{NL} \subseteq \mathbf{DSPACE}(\log^2 n)$ is a direct consequence of Theorem 2.1. The other inclusion follows by $\oplus\mathbf{L} \subseteq \mathbf{NC^2}$ (cf. [BDHM92]) and $\mathbf{NC^2} \subseteq \mathbf{DSPACE}(\log^2 n)$ (cf. [Vol99], Theorem 2.32). Note that $\mathbf{NC^2}$ is the class of all problems which can be solved by circuits of polynomial size and $\log^2 n$ depth (for details see [Vol99]).

Clearly any class of languages which can be solved by a deterministic Turing-machine is closed under complement (cf. [BDG95], Proposition 3.2). Hence the statement follows immediately for $\mathbf{L}$, $\mathbf{P}$, and $\mathbf{PSPACE}$. That $\mathbf{NL}$ is closed under complement is a direct consequence of Theorem 2.2. For the closure of $\oplus\mathbf{L}$ simply add one accepting path to the $\oplus\mathbf{L}$-computation (for details see [BDHM92]) and to show the closure of $\mathbf{PP}$ simply interchange the accepting and rejecting states of the used polynomial time Turing-machine (see [BDG95], Proposition 6.6).

For Statement 4 see [BDG95], Proposition 8.4 and Theorem 8.5.

For Statement 5 let $f \in \mathbf{FP}$. To show that $f \in \#\mathbf{P}$ simply compute $f(x)$ in polynomial time and branch nondeterministically into $f(x)$ accepting states (cf. [Val79]).

For the last statement let $f \in \#\mathbf{P}$ and M the corresponding polynomial time Turing-machine for f. Without loss of generality each path of M has length $p(|x|)$ for a suitable polynom p. Now let M' be the polynomial time Turing-machine that emerges if we interchange the accepting and rejecting states of M. Clearly $\#M = 2^{p(|x|)} - \#M'$. Now define $g(x) =_{\text{def}} 2^{p(|x|)}$ and $f'(x) =_{\text{def}} \#M'(x)$. $\square$

To compare the complexity of languages or functions we use *reducibilities*. For languages A and B we write $A \leq_{\text{m}}^{\text{log}} B$ if there exists a logspace computable function g such that $x \in A$ iff $g(x) \in B$ for all x. For functions f and h we write $f \leq_{\text{m}}^{\text{log}} h$ if there exists a logspace computable function g such that $f(x) = h(g(x))$ for all x. A function f is *logspace* 1*-Turing reducible* to h ($f \leq_{\text{1-T}}^{log} h$) if there exist two functions $g_1, g_2 \in \mathbf{FL}$ such that for all x we have: $f(x) = g_1(h(g_2(x)), x)$. We say that g_1, g_2 establish the 1-Turing reduction from f to h. When we claim a 1-Turing reducibility below, we will always witness this by constructing suitable functions g_1, g_2. All these reduction are reflexive and transitive relations.

A language A is called $\leq_{\text{m}}^{\text{log}}$*-hard* ($\leq_{\text{m}}^{\text{log}}$*-complete*, resp.) for a class $\mathcal{K}$ of languages iff $B \leq_{\text{m}}^{\text{log}} A$ for every $B \in \mathcal{K}$ ($A \in \mathcal{K}$ and $B \leq_{\text{m}}^{\text{log}} A$ for every $B \in \mathcal{K}$, resp.). In the same way one defines hardness and completeness for the reducibilities $\leq_{\text{m}}^{\text{log}}$ and $\leq_{\text{1-T}}^{log}$ for functions.

The next proposition will be used in varying contexts in this work.

Proposition 2.4 ([Sze87, HRV00]). *1.* $\mathbf{L}^{\mathbf{L}} = \mathbf{L}$
2. $\mathbf{L}^{\mathbf{NL}} = \mathbf{NL}$
3. $\mathbf{L}^{\oplus\mathbf{L}} = \oplus\mathbf{L}$

Proof. The first statement follows by the technique of recomputation. The second statement is a direct consequence of Theorem 2.2, because due to this result the logspace alternation hierarchy collapses to $\mathbf{NL}$. For the third statement see [HRV00]. $\square$

In this work we need the following problems, which are complete for the classes $\mathbf{NL}$ and $\oplus\mathbf{L}$, resp.

PROBLEM:	Graph Accessibility Problem (GAP)
INSTANCE:	A directed acyclic graph G whose vertices have outdegree 0 or 2, a start vertex s and a target vertex t
OUTPUT:	Is there a path in G which leads from s to t?

PROBLEM:	Graph Odd Accessibility Problem (GOAP)
INSTANCE:	A directed acyclic graph G whose vertices have outdegree 0 or 2, a start vertex s and a target vertex t
OUTPUT:	Is the number of paths in G, which lead from s to t, odd?

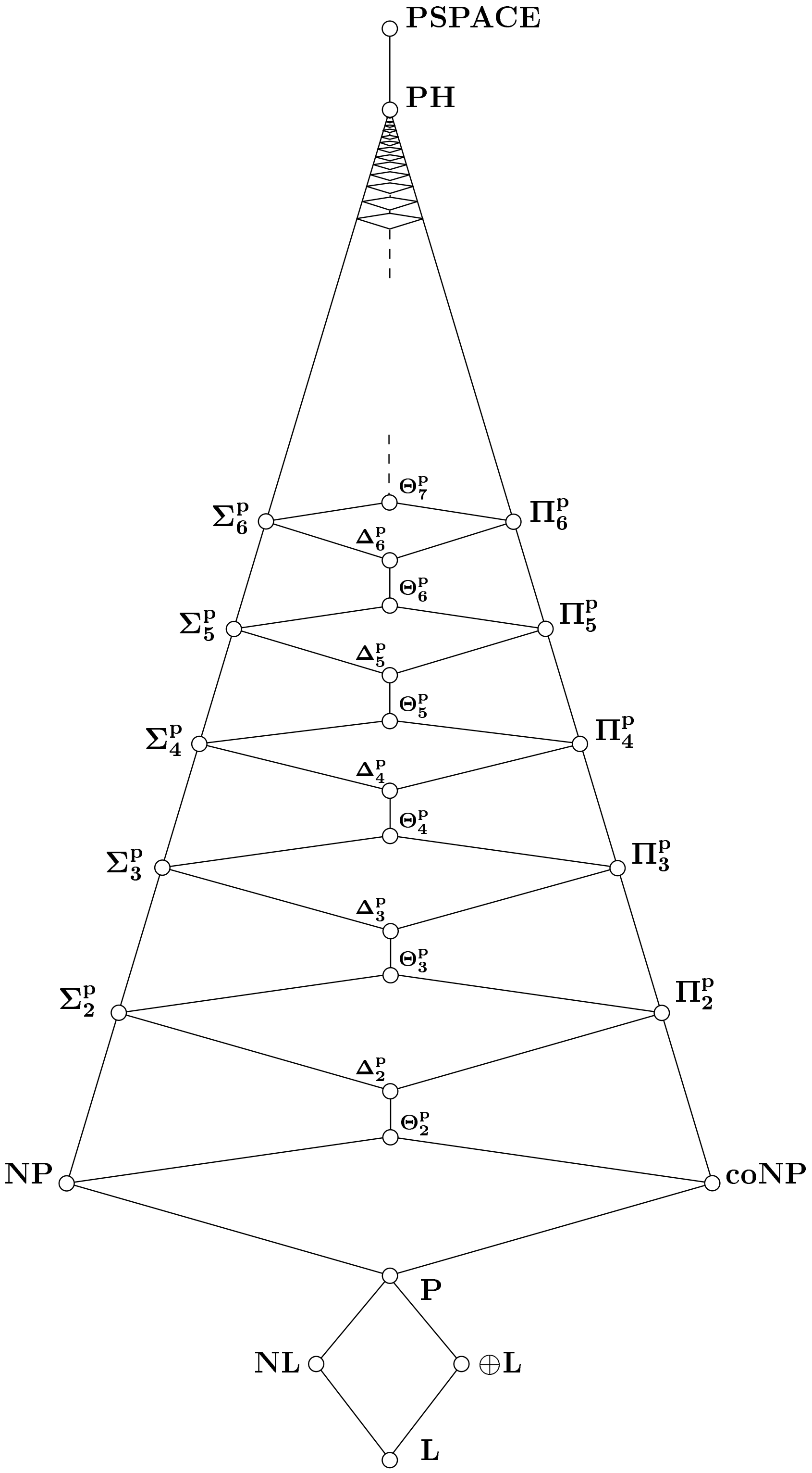


Fig. 2.1. The polynomial time hierarchy and some other important complexity classes.

Theorem 2.5. *1.* GAP *and* $\overline{\text{GAP}}$ *are* $\leq_{\mathrm{m}}^{\log}$*-complete for* **NL**.
2. GOAP *and* $\overline{\text{GOAP}}$ *are* $\leq_{\mathrm{m}}^{\log}$*-complete for* $\oplus$**L**.

Proof. Upper bounds. Obviously GAP $\in$ **NL** and GOAP $\in \oplus$**L**.
Lower bounds. Let M be a $O(\log n)$ space bounded Turing-machine, which accepts a language $A \in$ **NL** ($A \in \oplus$**L**, resp.).

We will call the complete description of the current state of M a *configuration* (see [Pap94], p. 21). Intuitively a configuration is a "snapshot" or "core-dump" of a Turing-machine while computing. So a configuration of M includes the current state, contents of the working tapes and the position of the heads on the working tape. Clearly a configuration of M needs at most $c \log n$ space for a suitable $c > 0$.

We can assume without loss of generality that M has exactly one accepting configuration and that every step of the computation nondeterministically branches exactly to two successor configurations. Moreover we can assume without loss of generality that M never reaches the same configuration two times. For this simply add a counter to M, which counts the performed steps during the computation.

Now we can construct in logarithmic space the *configuration graph* $G_{M(x)}$ of M on input x as follows: The set of vertices of $G_{M(x)}$ is the set of all possible configurations of M with input x. Two vertices C_1, C_2 of $G_{M(x)}$ are connected by a directed edge iff C_2 is a direct successor of C_1 in the computation of M on x. Note that because of the special nature of M each vertex of $G_{M(x)}$ has either outdegree 0 or 2 and $G_{M(x)}$ is clearly acyclic. Moreover we have a special start vertex s, the start configuration and a target vertex t, the accepting configuration.

Case 1: $A \in$ **NL**.
Clearly $x \in A$ iff M accepts x iff there is a path from s to t in $G_{M(x)}$ iff $(G_{M(x)}, s, t) \in$ GAP.
Case 2: $A \in \oplus$**L**.
Clearly $x \in A$ iff M accepts x iff there is a odd number of paths from s to t in $G_{M(x)}$ iff $(G_{M(x)}, s, t) \in$ GOAP.

This shows that for all $A \in$ **NL** ($A \in \oplus$**L**, resp.) it holds that $A \leq_{\mathrm{m}}^{\log}$ GAP ($A \leq_{\mathrm{m}}^{\log}$ GOAP, resp.). Since **NL** is closed under complement (see Theorem 2.3) also $\overline{\text{GAP}}$ is complete for **NL**. Additionally it is easy to show that $\oplus$**L** is closed under complement (see Theorem 2.3). Hence $\overline{\text{GOAP}}$ is complete for $\oplus$**L** too. □

Later we use the **coNP**-complete problem 3-TAUT, which is defined as follows:

PROBLEM: 3-TAUT
INSTANCE: A propositional formula H in 3-DNF
QUESTION: Is H a tautology?

In this work we will restrict ourselves to the case that each disjunct has exactly 3 literals.

Proposition 2.6 ([GJ79], LO8). 3-TAUT *is* $\leq_{\mathrm{m}}^{\log}$*-complete for* **coNP**.

2.2 Boolean functions and closed classes

A function $f\colon \{0,1\}^n \to \{0,1\}$ with $n \geq 0$ is called an *n-ary Boolean function*. By BF we denote the class of all Boolean functions. In particular, let 0 and 1 be the 0-ary constant functions having value 0 and 1, resp., let id and non be the unary functions defined by $\mathrm{id}(a) = a$ and $\mathrm{non}(a) = 1$ iff $a = 0$ and let et, vel, aeq, and aut be the binary functions defined by $\mathrm{et}(a,b) = 1$ iff $a = b = 1$, $\mathrm{vel}(a,b) = 0$ iff $a = b = 0$, $\mathrm{aeq}(a,b) = 1$ iff $a = b$, and $\mathrm{aut}(a,b) = 1$ iff $a \neq b$. We also write 0 instead of 0 (note that we use 0 as the symbol for the constant Boolean function 0), 1 instead of 1 (note that we use 1 as the symbol for the constant Boolean function 1), $\overline{x}$ or $\neg x$ instead of $\mathrm{non}(x)$, $x \wedge y$, $x \cdot y$, or xy instead of $\mathrm{et}(x,y)$, $x \vee y$ instead of $\mathrm{vel}(x,y)$, $x \leftrightarrow y$ instead of $\mathrm{aeq}(x,y)$, and $x \oplus y$ instead of $\mathrm{aut}(x,y)$. For $i \in \{1,\ldots,n\}$, the i-th variable of the n-ary Boolean function f is said to be *fictive* iff $f(a_1,\ldots a_{i-1},0,a_{i+1},\ldots,a_n) = f(a_1,\ldots a_{i-1},1,a_{i+1},\ldots,a_n)$ for all $a_1,\ldots a_{i-1},a_{i+1},\ldots,a_n \in \{0,1\}$.

For a set B of Boolean functions let $[B]$ be the smallest class which contains B and which is closed under superposition (i.e., substitution, permutation of variables and identification of variables, introduction of fictive variables). More precisely we define these operations as follows:

- *Substitution:* Let f^n and g^m be Boolean functions. Then we define h^{n+m-1} as $h(a_1,\ldots,a_{n-1},b_1,\ldots,b_m) =_{\mathrm{def}} f(a_1,\ldots,a_{n-1},g(b_1,\ldots,b_m))$ for all $a_1,\ldots,a_{n-1}$, $b_1,\ldots,b_m \in \{0,1\}$.
- *Permutation of variables:* Let f^n be a Boolean function and $\pi\colon \{1,\ldots,n\} \to \{1,\ldots,n\}$ be a permutation. Then we define $g(a_1,\ldots,a_n) =_{\mathrm{def}} f(a_{\pi(1)},\ldots,a_{\pi(n)})$ for all $a_1,\ldots,a_n \in \{0,1\}$.
- *Identification of the last variables:* Let f^n be a Boolean function. Then we define g^{n-1} as $g(a_1,\ldots,a_{n-1}) =_{\mathrm{def}} f(a_1,\ldots,a_{n-1},a_{n-1})$ for all $a_1,\ldots,a_{n-1} \in \{0,1\}$.
- *Introduction of a fictive variable:* Let f^n be a Boolean function. Then we define g^{n+1} as $g(a_1,\ldots,a_{n+1}) =_{\mathrm{def}} f(a_1,\ldots,a_n)$ for all $a_1,\ldots,a_{n+1} \in \{0,1\}$.

A set B of Boolean functions is called a *base* of the class F of Boolean functions if $[B] = F$ and it is called *complete* if $[B] = \mathrm{BF}$. A class F of Boolean functions is called *closed* if $[F] = F$.

We will see that the closed classes of Boolean functions are tightly related to the sets of Boolean functions computed by B-circuits and B-formulas (cf. Section 2.3).

Now consider some special properties of Boolean functions. Let $n \geq 0$. An n-ary Boolean function f is said to be

- *a-reproducing* iff $f(a,a,\ldots,a) = a$ $(a \in \{0,1\})$,
- *linear* iff there exist $a_0,a_1,\ldots,a_n \in \{0,1\}$ such that $f(b_1,\ldots,b_n) = a_0 \oplus (a_1 \wedge b_1) \oplus (a_2 \wedge b_2) \oplus \cdots \oplus (a_n \wedge b_n)$ for all $b_0,b_1,\ldots,b_n \in \{0,1\}$,
- *self-dual* iff $f(a_1,\ldots,a_n) = \overline{f(\overline{a_1},\ldots\overline{a_n})}$ for all $a_1,\ldots,a_n \in \{0,1\}$,
- *monotone* iff $f_m(a_1,\ldots,a_n) \leq f_m(b_1,\ldots,b_n)$ for all $a_1,\ldots,a_n,b_1,\ldots,b_n \in \{0,1\}$ such that $a_1 \leq b_1, a_2 \leq b_2,\ldots,a_n \leq b_n$,

- *a-separating* iff there exists an $i \in \{1,\dots,n\}$ such that $f^{-1}(a) \subseteq \{0,1\}^{i-1} \times \{a\} \times \{0,1\}^{n-i}$ ($a \in \{0,1\}$),
- *a-separating of degree m* iff for every $U \subseteq f^{-1}(a)$ such that $|U| = m$ there exists an $i \in \{1,\dots,n\}$ such that $U \subseteq \{0,1\}^{i-1} \times \{a\} \times \{0,1\}^{n-i}$ ($a \in \{0,1\}$, $m \geq 2$).

For a set B of Boolean functions we define B to be 0-reproducing, 1-reproducing, linear, self-dual, monotone, 0-separating, 1-separating, 0-separating of degree m or 1-separating of degree m if all functions $f \in B$ are 0-reproducing, 1-reproducing, linear, self-dual, monotone, 0-separating, 1-separating, 0-separating of degree m or 1-separating of degree m, resp.

The classes of all Boolean functions which are 0-reproducing, 1-reproducing, linear, self-dual, monotone, 0-separating, 1-separating, 0-separating of degree m, and 1-separating of degree m, resp., are denoted by R_0, R_1, L, D, M, S_0, S_1, S_0^m, and S_1^m, resp.

The closed classes of Boolean functions were intensively studied by E. L. Post already at the beginning of the twenties of the 20th century, where he gave a complete characterization of these classes (cf. [Pos41]). In this work we will make substantial use of his main results, which are presented in the following two theorems. For a detailed presentation see also the monograph [JGK70].

Theorem 2.7 ([Pos41]). *1. The complete list of closed classes of Boolean functions is:*
- BF, R_0, R_1, $\mathrm{R} =_{\mathrm{def}} \mathrm{R}_0 \cap \mathrm{R}_1$,
- M, $\mathrm{M}_0 =_{\mathrm{def}} \mathrm{M} \cap \mathrm{R}_0$, $\mathrm{M}_1 =_{\mathrm{def}} \mathrm{M} \cap \mathrm{R}_1$, $\mathrm{M}_2 =_{\mathrm{def}} \mathrm{M} \cap \mathrm{R}$,
- D, $\mathrm{D}_1 =_{\mathrm{def}} \mathrm{D} \cap \mathrm{R}$, $\mathrm{D}_2 =_{\mathrm{def}} \mathrm{D} \cap \mathrm{M}$,
- L, $\mathrm{L}_0 =_{\mathrm{def}} \mathrm{L} \cap \mathrm{R}_0$, $\mathrm{L}_1 =_{\mathrm{def}} \mathrm{L} \cap \mathrm{R}_1$, $\mathrm{L}_2 =_{\mathrm{def}} \mathrm{L} \cap \mathrm{R}$, $\mathrm{L}_3 =_{\mathrm{def}} \mathrm{L} \cap \mathrm{D}$,
- S_0, $\mathrm{S}_{02} =_{\mathrm{def}} \mathrm{S}_0 \cap \mathrm{R}$, $\mathrm{S}_{01} =_{\mathrm{def}} \mathrm{S}_0 \cap \mathrm{M}$, $\mathrm{S}_{00} =_{\mathrm{def}} \mathrm{S}_0 \cap \mathrm{R} \cap \mathrm{M}$,
- S_1, $\mathrm{S}_{12} =_{\mathrm{def}} \mathrm{S}_1 \cap \mathrm{R}$, $\mathrm{S}_{11} =_{\mathrm{def}} \mathrm{S}_1 \cap \mathrm{M}$, $\mathrm{S}_{10} =_{\mathrm{def}} \mathrm{S}_1 \cap \mathrm{R} \cap \mathrm{M}$,
- S_0^m, $\mathrm{S}_{02}^m =_{\mathrm{def}} \mathrm{S}_0^m \cap \mathrm{R}$, $\mathrm{S}_{01}^m =_{\mathrm{def}} \mathrm{S}_0^m \cap \mathrm{M}$, $\mathrm{S}_{00}^m =_{\mathrm{def}} \mathrm{S}_0^m \cap \mathrm{R} \cap \mathrm{M}$ *for* $m \geq 2$,
- S_1^m, $\mathrm{S}_{12}^m =_{\mathrm{def}} \mathrm{S}_1^m \cap \mathrm{R}$, $\mathrm{S}_{11}^m =_{\mathrm{def}} \mathrm{S}_1^m \cap \mathrm{M}$, $\mathrm{S}_{10}^m =_{\mathrm{def}} \mathrm{S}_1^m \cap \mathrm{R} \cap \mathrm{M}$ *for* $m \geq 2$,
- $\mathrm{E} =_{\mathrm{def}} [\mathrm{et}] \cup [0] \cup [1]$, $\mathrm{E}_0 =_{\mathrm{def}} [\mathrm{et}] \cup [0]$, $\mathrm{E}_1 =_{\mathrm{def}} [\mathrm{et}] \cup [1]$, $\mathrm{E}_2 =_{\mathrm{def}} [\mathrm{et}]$,
- $\mathrm{V} =_{\mathrm{def}} [\mathrm{vel}] \cup [0] \cup [1]$, $\mathrm{V}_0 =_{\mathrm{def}} [\mathrm{vel}] \cup [0]$, $\mathrm{V}_1 =_{\mathrm{def}} [\mathrm{vel}] \cup [1]$, $\mathrm{V}_2 =_{\mathrm{def}} [\mathrm{vel}]$,
- $\mathrm{N} =_{\mathrm{def}} [\mathrm{non}] \cup [0]$, $\mathrm{N}_2 =_{\mathrm{def}} [\mathrm{non}]$,
- $\mathrm{I} =_{\mathrm{def}} [\mathrm{id}] \cup [0] \cup [1]$, $\mathrm{I}_0 =_{\mathrm{def}} [\mathrm{id}] \cup [0]$, $\mathrm{I}_1 =_{\mathrm{def}} [\mathrm{id}] \cup [1]$, $\mathrm{I}_2 =_{\mathrm{def}} [\mathrm{id}]$,
- $\mathrm{C} =_{\mathrm{def}} [0] \cup [1]$, $\mathrm{C}_0 =_{\mathrm{def}} [0]$, $\mathrm{C}_1 =_{\mathrm{def}} [1]$ *and* $\emptyset$.

2. *All inclusional relationships between the closed classes of Boolean functions are presented in Figure 2.2.*
3. *There exists an algorithm which, given a finite set* $B \subseteq \mathrm{BF}$, *determines the closed class of Boolean functions from the list above which coincides with* $[B]$.
4. *There exists an algorithm which, given* $f \in \mathrm{BF}$ *and a finite set* $B \subseteq \mathrm{BF}$, *decides whether* $f \in [B]$ *or not.*

Now let us consider two examples:

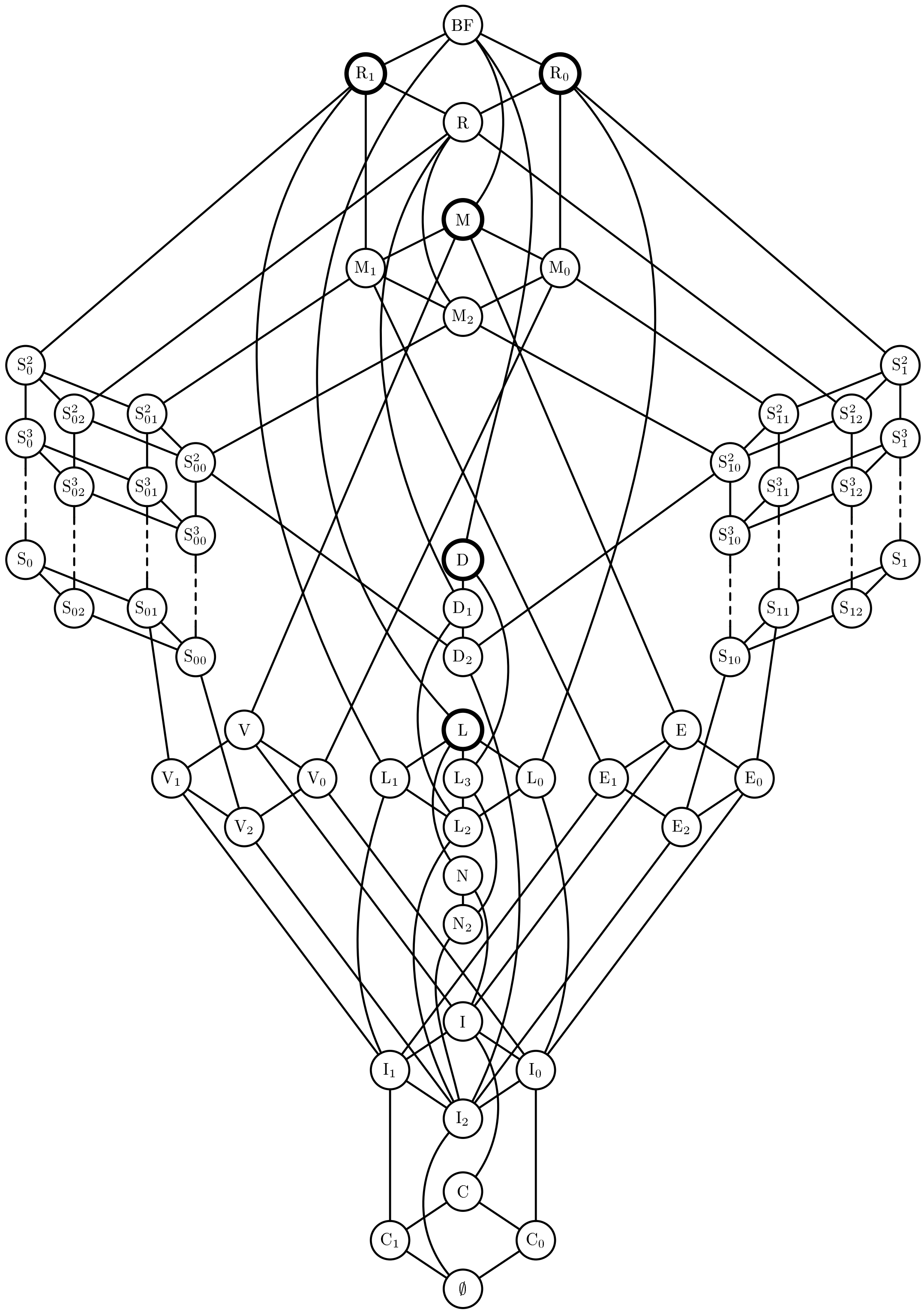


Fig. 2.2. Graph of all classes of Boolean functions being closed under superposition.

Example 2.8. We define the Boolean function f^3 such that $f(x,y,z) = 1$ iff exactly one argument has the value 1. Clearly, f is 0-reproducing but not 1-reproducing. Now assume that f is a linear function, i.e., $f(x,y,z) = a_0 \oplus (a_1 \wedge x) \oplus (a_2 \wedge y) \oplus (a_3 \wedge z)$ for suitable $a_0, a_1, a_2, a_3 \in \{0,1\}$. Since $f(0,0,0) = 0$ we know $a_0 = 0$ and it follows that $a_1 = a_2 = a_3 = 1$, because of $f(1,0,0) = f(0,1,0) = f(0,0,1) = 1$. But this is a contradiction because $f(1,1,1) = 0$, showing that f is not linear. Furthermore f is not monotone since $f(1,0,0) = 1$ and $f(1,1,1) = 0$, and it is not self-dual because of $f(0,0,0) = f(1,1,1)$. Finally, f cannot be 1-separating of degree 2 because of $f(0,0,1) = f(0,1,0) = 1$. Summarizing the above, f is not in R_1, L, M, D, and S_1^2 but it is in R_0. A short look at Figure 2.2 shows $[\{f\}] = \mathrm{R}_0$.

Example 2.9. Set $B =_{\text{def}} \{\text{vel}, g\}$, where $g(x,y) = \text{non}(\text{aut}(x,y))$. Now we want to determine $[B]$. Clearly B is not in R_0, since $g(0,0) = 1$, but in R_1, because $g(1,1) = \text{vel}(1,1) = 1$. The set B is not linear, because vel is not linear. For this suppose that vel is linear and therefore $\text{vel}(x,y) = a_0 \oplus (a_1 \wedge x) \oplus (a_2 \wedge y)$, where $a_0, a_1, a_2 \in \{0,1\}$. Hence $a_0 = 0$ and $a_1 = a_2 = 1$, because of $\text{vel}(0,0) = 0$ and $\text{vel}(0,1) = \text{vel}(1,0) = 1$. But this is a contradiction, since $\text{vel}(1,1) = 1$ too. Also B is not monotone, since $(0,0) < (1,0)$, but $g(0,0) = 1$ and $g(1,0) = 0$ and not selfdual, because $\text{vel}(0,1) = \text{vel}(1,0)$. Moreover B is not S_0^2, since $g^{-1}(0) = \{(0,1),(1,0)\}$. By Figure 2.2 we obtain that $[B] = \mathrm{R}_1$.

Because of Theorem 2.7 there exist five maximal non-complete closed classes of Boolean functions, which are circled bold in Figure 2.2. Using these classes the well-known result follows:

Corollary 2.10. *A class B of Boolean functions is complete if and only if $B \not\subseteq \mathrm{R}_0$, $B \not\subseteq \mathrm{R}_1$, $B \not\subseteq \mathrm{L}$, $B \not\subseteq \mathrm{D}$, and $B \not\subseteq \mathrm{M}$.*

The following theorem gives us some information about bases of closed classes of Boolean functions.

Theorem 2.11 ([Pos41]). *Every closed class of Boolean functions has a finite base. In particular:*

1. $\{\text{et}, \text{vel}, \text{non}\}$ *is a base of* BF.
2. $\{\text{vel}, x \wedge (y \oplus z \oplus 1)\}$ *is a base of* R.
3. $\{\text{et}, \text{aut}\}$ *is a base of* R_0.
4. $\{\text{vel}, x \oplus y \oplus 1\}$ *is a base of* R_1.
5. $\{\text{et}, \text{vel}, 0, 1\}$ *is a base of* M.
6. $\{\text{et}, \text{vel}, 1\}$ *is a base of* M_1.
7. $\{\text{et}, \text{vel}, 0\}$ *is a base of* M_0.
8. $\{\text{et}, \text{vel}\}$ *is a base of* M_2.
9. $\{(x \wedge \overline{y}) \vee (x \wedge \overline{z}) \vee (\overline{y} \wedge \overline{z})\}$ *is a base of* D.
10. $\{(x \wedge y) \vee (x \wedge \overline{z}) \vee (y \wedge \overline{z})\}$ *is a base of* D_1.
11. $\{(x \wedge y) \vee (y \wedge z) \vee (x \wedge z)\}$ *is a base of* D_2.
12. $\{\text{aut}, 1\}$ *is a base of* L.
13. $\{\text{aut}\}$ *is a base of* L_0.
14. $\{x \oplus y \oplus 1\}$ *is a base of* L_1.
15. $\{x \oplus y \oplus z\}$ *is a base of* L_2.
16. $\{x \oplus y \oplus z \oplus 1\}$ *is a base of* L_3.
17. $\{x \vee \overline{y}\}$ *is a base of* S_0.
18. $\{x \wedge \overline{y}\}$ *is a base of* S_1.

19. $\{x \vee (y \wedge \overline{z})\}$ *is a base of* S_{02}.
20. $\{x \wedge (y \vee \overline{z})\}$ *is a base of* S_{12}.
21. $\{x \vee (y \wedge z)\}$ *is a base of* S_{00}.
22. $\{x \wedge (y \vee z)\}$ *is a base of* S_{10}.
23. $\{0, x \wedge (y \vee z)\}$ *is a base of* S_{11}.
24. $\{1, x \vee (y \wedge z)\}$ *is a base of* S_{01}.
25. $\{x \vee \overline{y}, \mathrm{dual}(h_m)\}$ *is a base of* S_0^m.
26. $\{x \wedge \overline{y}, h_m\}$ *is a base of* S_1^m.
27. $\{x \vee (y \wedge \overline{z}), \mathrm{dual}(h_m)\}$ *is a base of* S_{02}^m.
28. $\{x \wedge (y \vee \overline{z}), h_m\}$ *is a base of* S_{12}^m.
29. $\{x \vee (y \wedge z), \mathrm{dual}(h_2)\}$ *is a base of* S_{00}^2.
30. $\{\mathrm{dual}(h_m)\}$ *is a base of* S_{00}^m, *where* $m \geq 3$.
31. $\{x \wedge (y \vee z), h_2\}$ *is a base of* S_{10}^2.
32. $\{h_m\}$ *is a base of* S_{10}^m, *where* $m \geq 3$.
33. $\{1, \mathrm{dual}(h_m)\}$ *is a base of* S_{01}^m.
34. $\{0, h_m\}$ *is a base of* S_{11}^m.
35. $\{\mathrm{et}, 0, 1\}$ *is a base of* E.
36. $\{\mathrm{et}, 1\}$ *is a base of* E_1.
37. $\{\mathrm{et}, 0\}$ *is a base of* E_0.
38. $\{\mathrm{et}\}$ *is a base of* E_2.
39. $\{\mathrm{vel}, 0, 1\}$ *is a base of* V.
40. $\{\mathrm{vel}, 1\}$ *is a base of* V_1.
41. $\{\mathrm{vel}, 0\}$ *is a base of* V_0.
42. $\{\mathrm{vel}\}$ *is a base of* V_2.
43. $\{\mathrm{non}, 1\}$ *is a base of* N.
44. $\{\mathrm{non}\}$ *is a base of* N_2.
45. $\{\mathrm{id}, 0, 1\}$ *is a base of* I.
46. $\{\mathrm{id}, 1\}$ *is a base of* I_1.
47. $\{\mathrm{id}, 0\}$ *is a base of* I_0.
48. $\{\mathrm{id}\}$ *is a base of* I_2.
49. $\{0, 1\}$ *is a base of* C.
50. $\{1\}$ *is a base of* C_1.
51. $\{0\}$ *is a base of* C_0.

We define $h_m(x_1, \ldots, x_{m+1}) =_{\mathrm{def}} \bigvee_{i=1}^{m+1} (x_1 \wedge \cdots \wedge x_{i-1} \wedge x_{i+1} \wedge \cdots \wedge x_{m+1})$.

For an n-ary Boolean function f define the Boolean function $\mathrm{dual}(f)$ by $\mathrm{dual}(f)(x_1, \ldots, x_n) =_{\mathrm{def}} \overline{f(\overline{x_1}, \ldots, \overline{x_n})}$. The functions f and $\mathrm{dual}(f)$ are said to be *dual*. Obviously, $\mathrm{dual}(\mathrm{dual}(f)) = f$. Furthermore, f is self-dual iff $\mathrm{dual}(f) = f$. For a class F of Boolean functions define $\mathrm{dual}(F) =_{\mathrm{def}} \{\mathrm{dual}(f) \mid f \in F\}$. The classes F and $\mathrm{dual}(F)$ are called *dual.*
For some examples of Boolean functions and their dual function see Figure 2.3.

f	$\mathrm{dual}(f)$
0	1
1	0
$\mathrm{et}(x, y)$	$\mathrm{vel}(x, y)$
$\mathrm{vel}(x, y)$	$\mathrm{et}(x, y)$
$\mathrm{non}(x)$	$\mathrm{non}(x)$
$\mathrm{id}(x)$	$\mathrm{id}(x)$
$\mathrm{aut}(x, y)$	$\mathrm{aeq}(x, y)$
$\mathrm{et}(x, \mathrm{non}(y))$	$\mathrm{vel}(x, \mathrm{non}(y))$

Fig. 2.3. Some Boolean functions and their dual function.

Proposition 2.12 (Duality principle). *1. Let g be a Boolean function such that* $g(x_1, \ldots, x_n) = f(f_1(x_1^1, \ldots, x_{m_1}^1), \ldots, f_l(x_1^l, \ldots, x_{m_l}^l))$. *For the dual function of g it holds that* $\mathrm{dual}(g) = \mathrm{dual}(f)(\mathrm{dual}(f_1), \ldots, \mathrm{dual}(f_l))$.
2. If B is a finite set of Boolean functions then $[\mathrm{dual}(B)] = \mathrm{dual}([B])$.

3. *Every closed class is dual to its "mirror class" (via the symmetry axis in Figure 2.2).*
4. *Let F be a closed class. Then* $\mathrm{dual}(F)$ *is a closed class too.*

Proof. For the first statement observe the following:

$$\begin{aligned}\mathrm{dual}(g) &= \mathrm{dual}(f(f_1(x_1^1,\dots,x_{m_1}^1),\dots,f_l(x_1^l,\dots,x_{m_l}^l)))\\ &= \neg f(f_1(\neg x_1^1,\dots,\neg x_{m_1}^1),\dots,f_l(\neg x_1^l,\dots,\neg x_{m_l}^l))\\ &= \neg f(\neg\neg f_1(\neg x_1^1,\dots,\neg x_{m_1}^1),\dots,\neg\neg f_l(\neg x_1^l,\dots,\neg x_{m_l}^l))\\ &= \neg f(\neg \mathrm{dual}(f_1(x_1^1,\dots,x_{m_1}^1)),\dots,\neg \mathrm{dual}(f_l(x_1^l,\dots,x_{m_l}^l)))\\ &= \mathrm{dual}(f)(\mathrm{dual}(f_1(x_1^1,\dots,x_{m_1}^1)),\dots,\mathrm{dual}(f_l(x_1^l,\dots,x_{m_l}^l)))\end{aligned}$$

The second statement is a direct consequence of the first statement.

For the third statement use the base B of an arbitrary closed class, given by Theorem 2.11, and calculate $\mathrm{dual}(B)$. Again by Theorem 2.11 we obtain that $\mathrm{dual}(B)$ is a base of the "mirror class", which proves the claim by the second statement.

Finally note that the last statement is also a direct consequence of the second statement. □

2.3 Generalized circuits and formulas

In the present work we focus on the study of computational problems related to generalized circuits and formulas, which are closely related to Post's framework (cf. Section 2.2). Hence we will introduce them in this section.

In basic textbooks propositional logic is always introduced in two steps. First we define the *syntax* by specifying propositional variables and connectors for combining them and the second step is the definition of the *semantic* of the defined formulas. In this step we associate a Boolean function to each connector. It is common to use $\wedge$, $\vee$ and $\neg$ as symbols for the Boolean functions et, vel and non, because $\{\mathrm{et}, \mathrm{vel}, \mathrm{non}\}$ is a complete set (cf. Corollary 2.10), and thus we can represent all Boolean functions by using formulas, which are build by making use of $\wedge$, $\vee$ and $\neg$ as connectors for propositional variables. A natural generalization would be the following approach. Choose a finite set of Boolean functions, not necessarily complete, and symbols for them. Now define formulas by using these symbols as connectors and their corresponding Boolean function as the semantic of them. By doing so, we can introduce for each fixed set B another generalized propositional logic. In a similar way we can introduce circuits. Here we only use gate-types which represent Boolean functions out of the fixed set of Boolean functions to build our circuits.

In the following chapters we will study various problems related to such generalized circuits and formulas, but due to the overwhelming number of them we have to restrict ourselves to a special subset of these problems. To have an easy to use definition we will first introduce our generalized circuits and after that we will consider the generalized formulas as a special case of them.

Informally, for a finite set B of Boolean functions, a *B-circuit C with input variables* $x_1, \ldots, x_n$ is a directed acyclic graph with a special output node, which has the following properties: Every vertex (gate) with in-degree 0 is labeled with an x_i or a 0-ary function from B. Every vertex (gate) with in-degree $k > 0$ is labeled with a k-ary function from B. Given values $a_1, \ldots, a_n \in \{0, 1\}$ to $x_1, \ldots, x_n$, every gate computes a Boolean value by applying the Boolean function of this gate to the values of the incoming edges. The Boolean value computed by the output gate is denoted by $f_C(a_1, \ldots, a_n)$. In such a way the B-circuit C computes the n-ary Boolean function f_C. To be more precise:

Definition 2.13. Let B be a finite set of Boolean functions and $\tilde{B}$ the set of symbols for them. A *B-circuit C with input-variables* $x_1, \ldots, x_n$ is a tuple

$$C(x_1, \ldots, x_n) = (P, E, o, \alpha, \beta, \gamma),$$

where (P, E) is a finite directed acyclic graph, $o \in P$ is the *output-gate*, $\alpha\colon P \to \{1, \ldots, |P|\}$ and $\gamma\colon E \to \{1, \ldots, |E|\}$ are injective functions and $\beta\colon P \to \tilde{B} \cup \{x_1, \ldots, x_n\}$ such that the following conditions hold:

- If $v \in P$ has in-degree 0, then $\beta(v) \in \{x_1, \ldots, x_n\}$ or $\beta(v)$ is a symbol for a 0-ary Boolean function from B.
- If $v \in P$ has in-degree $k > 0$, then $\beta(v)$ is a symbol for a k-ary Boolean function from B.

If $v \in P$ has in-degree k_0 and out-degree k_1, then we say: v is a *gate* in C with *fan-in* k_0 and *fan-out* k_1. If v is a gate in C we also write $v \in C$ instead of $v \in P$. If $e = (u, v) \in E$ then we say: e is a *wire* in C and u is a *predecessor-gate* of v. If $\beta(v) = x_i$ for some $1 \leq i \leq n$, then v is an *input-gate*. We denote the set of input-variables of C by $\mathrm{Var}(C) =_{\mathrm{def}} \{x_1, \ldots, x_n\}$.

Since a Boolean formula can be interpreted as a tree-like circuit, it is reasonable to define *B-formulas* as the subset of B-circuits C, such that each gate in C has at most fan-out 1.

The idea behind Definition 2.13 is that the function α gives each gate of C an unique number, that the function γ defines an ordering on the edges, and that the function β defines the *type* of $v \in P$, where v is either an input-gate or "computation gate", representing a Boolean function out of B.

Note that we explicitly allow that different input-gates are labeled by the same variable (i.e., β may not be injective). Where this property is useless in the circuit-case, we need it in the formula-case to make use of a variable for more than one time.

A B-circuit $C = (P, E, o, \alpha, \beta, \gamma)$ with n input-variables computes a Boolean function $f_C\colon \{0, 1\}^n \to \{0, 1\}$, as defined in the following:

Definition 2.14. Let B be a finite set of Boolean functions, $C = (P, E, o, \alpha, \beta, \gamma)$ be a B-circuit with input-variables $x_1, \ldots, x_n$, and $a_1, \ldots, a_n \in \{0, 1\}$. Now we define for each gate v of C inductively a function $f_v\colon \{0, 1\}^n \to \{0, 1\}$ as follows:

- If $v \in P$ has fan-in 0 and $\beta(v) = x_i$, where $1 \leq i \leq n$, then $f_v(a_1, \ldots, a_n) =_{\text{def}} a_i$. If v has fan-in 0 and $\beta(v) = c$ is a symbol for a 0-ary function from B, then $f_v(a_1, \ldots, a_n) =_{\text{def}} c$.
- If $v \in P$ has fan-in $k > 0$, $\beta(v)$ is a symbol for a k-ary function out of B, $v_1, \ldots, v_k$ are predecessor-gates of v which are ordered by $\gamma(v_1) < \cdots < \gamma(v_k)$ and the wire (v_i, v) represents a substitution of the i-th argument of $f_{\beta(v)}$ by $f_{v_i}(a_1, \ldots, a_n)$, then $f_v(a_1, \ldots, a_n) =_{\text{def}} f_{\beta(v)}(f_{v_1}(a_1, \ldots, a_n), \ldots, f_{v_k}(a_1, \ldots, a_n))$, where $f_{\beta(v)}$ is the Boolean function represented by $\beta(v)$.

Now we define the function computed by C as $f_C(a_1, \ldots, a_n) =_{\text{def}} f_o(a_1, \ldots, a_n)$.

Additionally note that not all input-gates must be connected to other gates in a B-circuit or B-formula, because we allow unconnected input-gates, having a fan-out 0 and fan-in 0. Clearly these variables are fictive, and we will call them *syntactically fictive variables*. Additionally our B-formulas have the symbol for the id-function for free, because we are allowed to introduce wires independently from the used set B. By using the definitions above we obtain the following obvious proposition:

Proposition 2.15. *Let B be a finite set of Boolean functions. Then*

$$\{f_H \mid H \textit{ is } B\textit{-formula}\} = [B \cup \{\text{id}\}] = \{f_C \mid C \textit{ is } B\textit{-circuit}\}.$$

Furthermore, let $\langle B \rangle =_{\text{def}} \{f_C \mid C \text{ is a } B\text{-circuit}\}$ be the set of all Boolean functions which can be computed by B-circuits.

Finally have in mind that the term *gate-type* will be replaced by *function-symbol* when we work with B-formulas. This is done for more closeness to the terms used in (propositional) logic.

Definition 2.16. Let B be a finite set of Boolean functions and C (H, resp.) a B-circuit (B-formula, resp.). Then $\text{dual}(C)$ ($\text{dual}(H)$, resp.) is the $\text{dual}(B)$-circuit ($\text{dual}(B)$-formula, resp.), which arises from C (H, resp.), when we replace every gate of type f (f-function symbol, resp.) by a gate of type $\text{dual}(f)$ ($\text{dual}(f)$-function symbol, resp.).

By this definition we directly obtain that $f_{\text{dual}(C)} = \text{dual}(f_C)$ (cf. Proposition 2.12.1).

Let $V = \{x_1, \ldots, x_n\}$ be a finite set of propositional variables. An *assignment with respect to* V is a function $I\colon V \to \{0, 1\}$. When the set V of variables is clear from the context, we will simply speak of an *assignment*. In order for an assignment w.r.t. V to be compatible with a circuit C (formula H, resp.), we must have $\text{Var}(C) = V$ ($\text{Var}(H) = V$, resp.). An assignment I *satisfies* a circuit $C(x_1, \ldots, x_n)$ (a formula $H(x_1, \ldots, x_n)$, resp.), if $f_C(I(x_1), \ldots, I(x_n)) = 1$ ($f_H(I(x_1), \ldots, I(x_n)) = 1$, resp.). That an assignment I satisfies C (H, resp.) will be denoted by $I \models C$ ($I \models H$, resp.). We will write $C_1(x_1, \ldots, x_n) \equiv C_2(x_1, \ldots, x_n)$ for B-circuits C_1 and C_2, if $f_{C_1}(a_1, \ldots, a_n) = f_{C_2}(a_1, \ldots, a_n)$ for all

$a_1, \dots, a_n \in \{0,1\}$. An analogous notation will be used for B-formulas. If I is an assignment w.r.t. V, $y \notin V$, and $a \in \{0,1\}$, then $I \cup \{y := a\}$ denotes the assignment I' w.r.t. $V \cup \{y\}$, defined by $I'(y) = a$ and $I'(x) = I(x)$ for all $x \neq y$. On the other hand, for $\{x_{i_1}, x_{i_2}, \dots, x_{i_m}\} \subseteq V$, we denote by $I' = I/\{x_{i_1}, x_{i_2}, \dots, x_{i_m}\}$ the assignment I' w.r.t. $\{x_{i_1}, x_{i_2}, \dots, x_{i_m}\}$, given by $I'(x) = I(x)$ iff $x \in \{x_{i_1}, x_{i_2}, \dots, x_{i_m}\}$. If $V = \{x_1, \dots, x_k\}$, $x_1 < \dots < x_k$, then an assignment I with $I(x_i) = a_i$ will also be denoted by $(a_1, \dots, a_k)$.

For counting the number of satisfying and unsatisfying assignments of B-circuits (B-formulas, resp.) we define a counting operator as follows:

Definition 2.17. Let B be a finite set of Boolean functions and C be a B-circuit. We define $\#_1(C(x_1, \dots, x_n)) =_{\text{def}} |\{\alpha \in \{0,1\}^n \mid f_C(\alpha) = 1\}|$ and similarly $\#_0(C(x_1, \dots, x_n)) =_{\text{def}} |\{\alpha \in \{0,1\}^n \mid f_C(\alpha) = 0\}|$.

An analogous definition will be used for B-formulas.

Proposition 2.18. *Let B be a finite set of Boolean functions and C be a B-circuit. Then the following equality holds for B-circuits:* $\#_1(C(x_1, \dots, x_n)) = 2^n - \#_1(\text{dual}(C(x_1, \dots, x_n)))$.

Proof. Follows by the simple fact that $\#_1(C) = \#_0(\text{dual}(C))$. □

Note that the same statement holds for B-formulas too.

Now let $\mathbf{A}$ be a property related to Boolean functions. For a finite set B of Boolean functions define

$$\mathbf{A}(B) =_{\text{def}} \{(C, a) \mid C \text{ is a } B\text{-circuit such that } (f_C, a) \text{ has property } \mathbf{A}\}.$$

In order to study the complexity of $\mathbf{A}(B)$ the following question is very important: How can we relate the complexity of $\mathbf{A}(B)$ and $\mathbf{A}(B')$ by relating the sets B and B' of Boolean functions? The following proposition gives a satisfactory answer.

Proposition 2.19. *Let $\mathbf{A}$ be a property of Boolean functions, and let B and B' be finite sets of Boolean functions.*

1. *If $B \subseteq \langle B' \rangle$ then $\mathbf{A}(B) \leq_{\text{m}}^{\log} \mathbf{A}(B')$.*
2. *If $\langle B \rangle = \langle B' \rangle$ then $\mathbf{A}(B) \equiv_{\text{m}}^{\log} \mathbf{A}(B')$.*

Proof. Statement 2 is an immediate consequence of Statement 1. For the latter just mention the obvious fact that $B \subseteq \langle B' \rangle$ implies that there exists a logspace computable function g converting every B-circuit C into a B'-circuit $g(C)$ such that $f_C = f_{g(C)}$. (A gate which represents h in C is simply replaced by a B'-circuit computing h.) □

Note that a similar proposition cannot be stated for B-formulas easily. The reason for this is that replacing each function-symbol in a B-formula H by a suitable B'-formula can result in an exponential sized B'-formula $g(H)$. Clearly this cannot be done by a logspace function (cf. Sec 3.7).

In the following we will make substantial use of the fact that the complexity of $\mathbf{A}(B)$ depends only on $\langle B\rangle$. To this end it is necessary to study the classes of Boolean functions which have the form $\langle B\rangle$. As seen above, it turns out that these are exactly those classes of Boolean functions which contain the identity function and which are closed under superposition (i.e., substitution, permutation of variables, identification of variables, and introduction of fictive variables). Because of this connection we can make heavy use of Post's results in the following.

3. Satisfiability- and counting-problems

3.1 Introduction

The complexity of formula-based and circuit-based combinatorial problems was studied through more than three decades of Complexity Theory. Already in 1971, S.A. Cook [Coo71] proved that the satisfiability problem for Boolean formulas is **NP**-complete: This was the first **NP**-complete problem ever discovered. R.E. Ladner [Lad77] proved in 1977 that the circuit value problem is **P**-complete. In many cases when a new complexity class was introduced and investigated, a formula-based or circuit-based combinatorial problem was the first which was proven to be complete for this class ([SM73, Gil77], for example). However, usually these problems were defined using formulas or circuits with a complete base of function symbols or gates, mostly with the base $\{\wedge, \vee, \neg\}$. But what can be said about the complexity of such problems when a different base is used? There are several special results of this kind (e.g. [Sim75, Gol77, Lew79, GP86]). In particular, there are very detailed investigations for the special case of Boolean formulas in conjunctive normal form in [Sch78, Cre95, CH96, CH97, KST97, RV00]. But there are no results answering this question for generalized circuits and formulas in full generality. In this chapter we will give complete characterizations of the complexity of some combinatorial problems defined by B-circuits and B-formulas

In Sections 3.2-3.6 we study the complexity of the circuit value problem, the satisfiability problem and the tautology problem, some quantified circuit problems, the counting function, and the threshold problem for B-circuits. In Section 3.7 we end up with some results about B-formulas, i.e., tree-like B-circuits. We give complete characterizations of their complexity as in the following example. Let the satisfiability problem $\mathrm{SAT}^{\mathrm{C}}(B)$ be the set of all B-circuits which evaluate to 1 for at least one input tuple. If B consists only of 1-reproducing functions, of self-dual functions, or of non-functions (for definitions see Section 2.2), then $\mathrm{SAT}^{\mathrm{C}}(B)$ is in **L**. Otherwise, if B consists only of and-functions or of or-functions then $\mathrm{SAT}^{\mathrm{C}}(B)$ is $\leq_{\mathrm{m}}^{\log}$-complete for **NL**. Otherwise, if B consists only of linear functions then $\mathrm{SAT}^{\mathrm{C}}(B)$ is $\leq_{\mathrm{m}}^{\log}$-complete for $\oplus$**L**. Otherwise, if $\mathrm{SAT}^{\mathrm{C}}(B)$ consists only of monotone functions then $\mathrm{SAT}^{\mathrm{C}}(B)$ is $\leq_{\mathrm{m}}^{\log}$-complete for **P**. Otherwise $\mathrm{SAT}^{\mathrm{C}}(B)$ is $\leq_{\mathrm{m}}^{\log}$-complete for **NP**.

Note that the results of this chapter are based on [RW00].

3.2 The circuit value problem

Let B be a finite set of Boolean functions. The *circuit value problem for B-circuits* is defined as:

PROBLEM: $\mathrm{VAL}^{\mathrm{C}}(B)$
INSTANCE: A B-circuit $C(x_1, \ldots, x_n)$ and $a \in \{0,1\}^n$
QUESTION: Is $f_C(a) = 1$?

Similarly we define the *formula value problem for B-formulas* $\mathrm{VAL}^{\mathrm{F}}(B)$.

It is obvious that $\mathrm{VAL}^{\mathrm{C}}(B) \in \mathbf{P}$ for every finite set B of Boolean functions. The polynomial time function *eval* from Figure 3.2 illustrates this simple fact.

```
function eval(C(x_1, ..., x_n), (a_1, ..., a_n));
begin
      mark all gates which are labeled by x_i with a_i;
      mark all gates associated with the constant 0 (1, resp.) function by 0 (1, resp.);
      while (output-gate is not marked) do begin
            pick an unmarked gate g(x_1, ..., x_n) whose predecessors p_1, ..., p_n
            are marked by b_1, ..., b_n;
            calculate c = f_g(b_1, ..., b_n); /* Function f_g is associated with gate g */
            mark the gate g with c
      end;
      if (output-gate is marked with 1) then
         accept
      else
         reject
end.
```

Fig. 3.1. A polynomial-time algorithm for $\mathrm{VAL}^{\mathrm{C}}(B)$.

The following facts on the complexity of $\mathrm{VAL}^{\mathrm{C}}(B)$ are well known from the literature.

Theorem 3.1. *Let B be a finite set of Boolean functions.*

1. *[Lad77] If $[B] = \mathrm{BF}$ then $\mathrm{VAL}^{\mathrm{C}}(B)$ is $\leq_{\mathrm{m}}^{\log}$-complete for $\mathbf{P}$.*
2. *[Gol77] If $\{\mathrm{et}, \mathrm{vel}\} \subseteq [B]$ then $\mathrm{VAL}^{\mathrm{C}}(B)$ is $\leq_{\mathrm{m}}^{\log}$-complete for $\mathbf{P}$.*
3. *[GP86] Let B be a set of binary Boolean functions. If $\{\mathrm{et}, \mathrm{vel}\} \subseteq [B]$ or ($B \not\subseteq \mathrm{L}$ and $B \not\subseteq \mathrm{M}$) then $\mathrm{VAL}^{\mathrm{C}}(B)$ is $\leq_{\mathrm{m}}^{\log}$-complete for $\mathbf{P}$, otherwise $\mathrm{VAL}^{\mathrm{C}}(B)$ is acceptable in $\log^2 n$ space.*

We start the investigation of the complexity of $\mathrm{VAL}^{\mathrm{C}}(B)$ by strengthening Proposition 2.19 for this particular case.

Proposition 3.2. *Let B and B' be finite sets of Boolean functions.*

1. *If $B \subseteq [B' \cup \{\mathrm{id}, 0, 1\}]$ then $\mathrm{VAL}^{\mathrm{C}}(B) \leq_{\mathrm{m}}^{\log} \mathrm{VAL}^{\mathrm{C}}(B')$.*
2. *If $[B \cup \{\mathrm{id}, 0, 1\}] = [B' \cup \{\mathrm{id}, 0, 1\}]$ then $\mathrm{VAL}^{\mathrm{C}}(B) \equiv_{\mathrm{m}}^{\log} \mathrm{VAL}^{\mathrm{C}}(B')$.*

Proof. As for Proposition 2.19, but additionally a 0-gate (1-gate, resp.) is replaced by an input gate labeled with the Boolean value 0 (1, resp.). □

Therefore, for the study of the complexity of $\mathrm{VAL^C}(B)$, only those closed classes of Boolean functions are of importance which contain id, 0, and 1. To obtain Proposition 3.3 take for any closed class of Figure 2.2 the corresponding base F, given in Theorem 2.11 and calculate $[F \cup \{0, 1, \mathrm{id}\}]$.

Proposition 3.3. *The closed classes of Boolean functions containing* id, 0, *and* 1 *are* BF, M, V, E, L, N, *and* IC, *where* $\mathrm{IC} =_{\mathrm{def}} [\mathrm{id}, 0, 1]$. *The inclusional relationships between these classes are presented in Figure 3.2.*

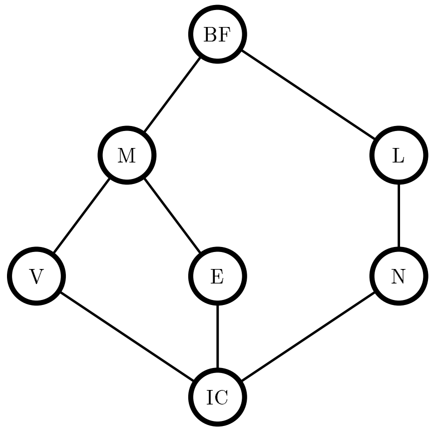


Fig. 3.2. All closed classes of Boolean functions containing id, 0, and 1.

Now we are ready to prove our main theorem on the complexity of $\mathrm{VAL^C}(B)$.

Theorem 3.4. *Let B be a finite set of Boolean functions:*

if *($B \subseteq \mathrm{N}$)* ***then***
$\mathrm{VAL^C}(B) \in \mathbf{L}$
else if *(($B \subseteq \mathrm{E}$)* ***or*** *($B \subseteq \mathrm{V}$))* ***then***
$\mathrm{VAL^C}(B)$ *is* $\leq_{\mathrm{m}}^{\log}$*-complete for* $\mathbf{NL}$
else if *($B \subseteq \mathrm{L}$)* ***then***
$\mathrm{VAL^C}(B)$ *is* $\leq_{\mathrm{m}}^{\log}$*-complete for* $\oplus\mathbf{L}$
else
$\mathrm{VAL^C}(B)$ *is* $\leq_{\mathrm{m}}^{\log}$*-complete for* $\mathbf{P}$

There exists an algorithm which decides which of the cases above takes place.

Proof. 1. If $B \subseteq \mathrm{N} = [\{\mathrm{non}, \mathrm{id}, 0, 1\}]$ then, by Proposition 3.2, we get $\mathrm{VAL^C}(B)$ $\leq_{\mathrm{m}}^{\log}$ $\mathrm{VAL^C}(\{\mathrm{non}\})$. For a $\{\mathrm{non}\}$-circuit C, there is $f_C(a) = 1$ iff the non-fictive path in backward direction, starting at the output-gate, either has even length and ends in an input-gate with value 1 or has odd length and ends in an input-gate with value 0. This can be checked in logarithmic space.

2. Let $B \not\subseteq \mathrm{N}$ and $B \subseteq \mathrm{V}$. By Proposition 3.3 and Theorem 2.11 we obtain $[B \cup \{\mathrm{id}, 0, 1\}] = \mathrm{V} = [\{\mathrm{vel}, \mathrm{id}, 0, 1\}]$, and by Proposition 3.2 we get $\mathrm{VAL^C}(B) \equiv_{\mathrm{m}}^{\log} \mathrm{VAL^C}(\{\mathrm{vel}\})$.
Upper bound. For a $\{\mathrm{vel}\}$-circuit C there is $f_C(a) = 1$ iff at least one of the non-fictive paths in backward direction from the output-gate ends in an input-gate with value 1. This is a graph accessibility problem which can be solved in nondeterministic logarithmic space.
Lower bound. We prove $\mathrm{GAP} \leq_{\mathrm{m}}^{\log} \mathrm{VAL^C}(\{\mathrm{vel}\})$ (cf. Theorem 2.5.1). Given a directed acyclic graph G whose vertices have outdegree 0 or 2, a start vertex s, and a target vertex t with outdegree 0. We construct in logarithmic space a $\{\mathrm{vel}\}$-circuit C as follows: Every outdegree 2 vertex becomes a vel-gate, the vertex t becomes an input gate with input value 1, and every other outdegree 0 vertex becomes an input gate with input value 0. The vertex s becomes the output-gate of C. Let a be the input string constructed (i.e., the input-values of all input-gates) in this way. Obviously, there exists a path in G from s to t iff $(C, a) \in \mathrm{VAL^C}(B)$.

3. The case $B \not\subseteq \mathrm{N}$ and $B \subseteq \mathrm{E}$ is treated in the same way; just replace V by E, vel by et, GAP by $\overline{\mathrm{GAP}}$ and mark t with input-value 0 and every other outdegree 0 vertex with input-value 1.

4. Let $B \not\subseteq \mathrm{N}$ and $B \subseteq \mathrm{L}$. By Proposition 3.3 and Theorem 2.11 we obtain $[B \cup \{\mathrm{id}, 0, 1\}] = \mathrm{L} = [\{\mathrm{aut}, \mathrm{id}, 0, 1\}]$, and by Proposition 3.2 we get $\mathrm{VAL^C}(B) \equiv_{\mathrm{m}}^{\log} \mathrm{VAL^C}(\{\mathrm{aut}\})$.
Upper bound. For an $\{\mathrm{aut}\}$-circuit C there is $f_C(a) = 1$ iff the number of non-fictive paths from the output-gate to an input-gate with value 1 is odd. This can be checked by a $\oplus\mathbf{L}$-computation.
Lower bound. We prove $\mathrm{GOAP} \leq_{\mathrm{m}}^{\log} \mathrm{VAL^C}(\{\mathrm{aut}\})$ (cf. Theorem 2.5.2). This can be done in the same way as $\mathrm{GAP} \leq_{\mathrm{m}}^{\log} \mathrm{VAL^C}(\{\mathrm{vel}\})$ in Item 2 of this proof.

5. Let $B \not\subseteq \mathrm{V}$, $B \not\subseteq \mathrm{E}$, and $B \not\subseteq \mathrm{L}$. By Proposition 3.3 we have $\{\mathrm{et}, \mathrm{vel}\} \subseteq \mathrm{M} \subseteq [B \cup \{\mathrm{id}, 0, 1\}]$, and by Proposition 3.2 we get $\mathrm{VAL^C}(\{\mathrm{et}, \mathrm{vel}\}) \leq_{\mathrm{m}}^{\log} \mathrm{VAL^C}(B)$. However, $\mathrm{VAL^C}(\{\mathrm{et}, \mathrm{vel}\})$ is $\leq_{\mathrm{m}}^{\log}$-complete for $\mathbf{P}$ by Theorem 3.1, and $\mathrm{VAL^C}(B)$ is obviously in $\mathbf{P}$. □

Notice that Theorem 3.4 improves the Goldschlager-Parberry result (Theorem 3.1.3) in two ways:

- Theorem 3.1.3 applies only to finite sets B of *binary* Boolean functions, whereas Theorem 3.4 applies to all finite sets B of Boolean functions.
- Theorem 3.1.3 distinguishes only between "$\mathbf{P}$-complete" and "acceptable in $\log^2 n$ space", whereas Theorem 3.4 splits the latter case into the cases "$\mathbf{NL}$-complete", "$\oplus\mathbf{L}$-complete", and "acceptable in $\log n$ space" (cf. Theorem 2.3).

Finally let us mention that the Goldschlager-Parberry criterion for sets B of binary Boolean functions is not valid for arbitrary finite sets B of Boolean functions. Take for example $B = \{xy \vee xz\}$. By Theorem 2.11 we know that $[B] = \mathrm{S}_{10}$ and $[\{\mathrm{vel}\}] = \mathrm{V}_2$. A short look at Figure 2.2 shows that $\mathrm{vel} \notin [B]$. Because of $B \subseteq \mathrm{M}$ this would correspond to the "otherwise" case of Theorem 3.1.3. However, Theorem 3.4 shows that $\mathrm{VAL^C}(B)$ is $\leq_{\mathrm{m}}^{\log}$-complete for $\mathbf{P}$.

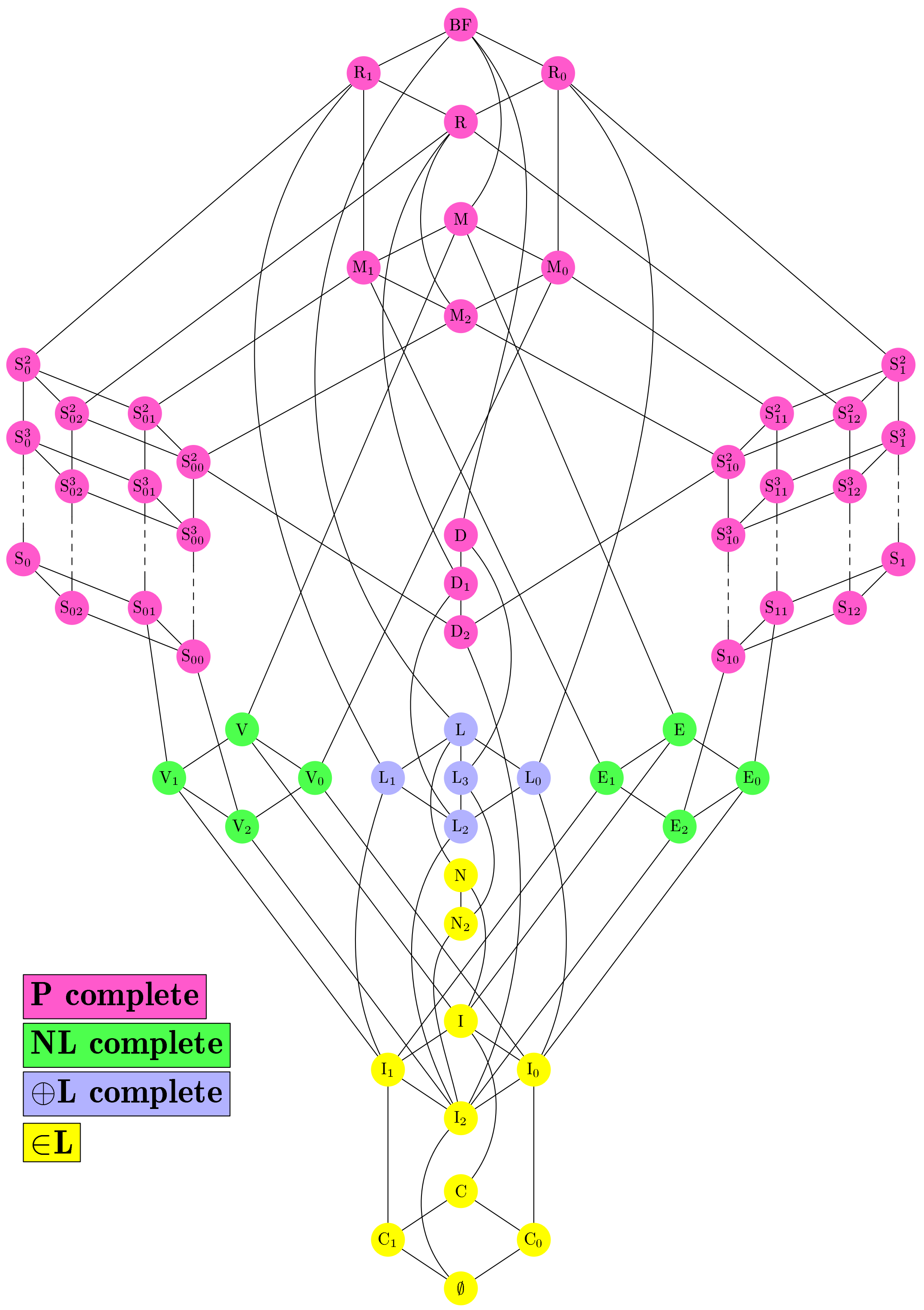


Fig. 3.3. The complexity of $\mathrm{VAL}^{\mathrm{C}}(B)$.

3.3 Satisfiability and tautology

In this section we study the complexity of the satisfiability problem and the tautology problem for B-circuits. For a finite set B of Boolean functions we define:

PROBLEM: $\mathrm{SAT}^{\mathrm{C}}(B)$
INSTANCE: A B-circuit $C(x_1, \dots, x_n)$
QUESTION: Exists an $a \in \{0,1\}^n$ such that $f_C(a) = 1$?

PROBLEM: $\mathrm{TAUT}^{\mathrm{C}}(B)$
INSTANCE: A B-circuit $C(x_1, \dots, x_n)$
QUESTION: For all $a \in \{0,1\}^n$ hold $f_C(a) = 1$?

It is obvious that $\mathrm{SAT}^{\mathrm{C}}(B) \in \mathbf{NP}$ and $\mathrm{TAUT}^{\mathrm{C}}(B) \in \mathbf{coNP}$ for every finite set B of Boolean functions. The following facts about the complexity of $\mathrm{SAT}^{\mathrm{C}}(B)$ can easily be derived from the literature, where special cases of B-formulas (i.e., B-circuits with fan-out at most 1) were considered.

Theorem 3.5. *Let B be a finite set of Boolean functions.*

1. *[Coo71] If $[B] = \mathrm{BF}$ then $\mathrm{SAT}^{\mathrm{C}}(B)$ is $\leq_{\mathrm{m}}^{\log}$-complete for* $\mathbf{NP}$ *and $\mathrm{TAUT}^{\mathrm{C}}(B)$ is $\leq_{\mathrm{m}}^{\log}$-complete for* $\mathbf{coNP}$.
2. *[Lew79] If $x \wedge \overline{y} \in [B]$ then $\mathrm{SAT}^{\mathrm{C}}(B)$ is $\leq_{\mathrm{m}}^{\log}$-complete for* $\mathbf{NP}$.

In some special cases the complexity of $\mathrm{SAT}^{\mathrm{C}}(B)$ can be related to the complexity of $\mathrm{VAL}^{\mathrm{C}}(B)$.

Proposition 3.6. *Let B be a finite set of Boolean functions.*

1. *If $B \subseteq \mathrm{M}$ then $\mathrm{SAT}^{\mathrm{C}}(B) \leq_{\mathrm{m}}^{\log} \mathrm{VAL}^{\mathrm{C}}(B)$.*
2. *If $0 \in [B] \subseteq \mathrm{M}$ then $\mathrm{VAL}^{\mathrm{C}}(B) \leq_{\mathrm{m}}^{\log} \mathrm{SAT}^{\mathrm{C}}(B)$.*
3. *If $0, 1 \in [B]$ then $\mathrm{VAL}^{\mathrm{C}}(B) \leq_{\mathrm{m}}^{\log} \mathrm{SAT}^{\mathrm{C}}(B)$.*

Proof. 1. The reduction is given by $C(x_1, \dots, x_n) \in \mathrm{SAT}^{\mathrm{C}}(B)$ iff $(C, 1^n) \in \mathrm{VAL}^{\mathrm{C}}(B)$.

2. Convert a B-circuit $C(x_1, \dots, x_n)$ with an input string $a \in \{0,1\}^n$ into a B-circuit $C'(x_1, \dots, x_n)$ as follows: Replace every input gate with value 0 by a B-circuit computing 0 and replace every input gate with value 1 by an input gate with a new input variable. Obviously, $(C, a) \in \mathrm{VAL}^{\mathrm{C}}(B)$ iff $C' \in \mathrm{SAT}^{\mathrm{C}}(B)$, because f_C is a monotone function.

3. Convert a B-circuit $C(x_1, \dots, x_n)$ with an input string $a \in \{0,1\}^n$ into a B-circuit C' as follows: Replace every input gate with value 0 by a B-circuit computing 0 and replace every input gate with value 1 by a B-circuit computing 1. Obviously, $(C, a) \in \mathrm{VAL}^{\mathrm{C}}(B)$ iff $C' \in \mathrm{SAT}^{\mathrm{C}}(B)$. □

Proposition 3.7. *If B is a finite set of Boolean functions containing* et *then $\mathrm{SAT}^{\mathrm{C}}(B \cup \{1\}) \leq_{\mathrm{m}}^{\log} \mathrm{SAT}^{\mathrm{C}}(B)$.*

Proof. A $(B\cup\{1\})$-circuit C with input variables $x_1,\dots,x_n$ is converted into a B-circuit C' with input variables $x_1,\dots,x_n,z$ by replacing every 1-gate by a z-gate. Hence $f_{C'}(x_1,\dots,x_n,1) = f_C(x_1,\dots,x_n)$. Now construct a B-circuit C'' with input variables $x_1,\dots,x_n,z$ such that $f_{C''}(x_1,\dots,x_n,z) = f_{C'}(x_1,\dots,x_n,z)\wedge z$. We obtain: C is satisfiable iff C'' is satisfiable. □

Now we are ready to prove the main theorem on the complexity of $\mathrm{SAT}^{\mathrm{C}}(B)$.

Theorem 3.8. *Let B be a finite set of Boolean functions.*

if *$((B\subseteq \mathrm{R}_1)$ **or** $(B\subseteq \mathrm{D})$ **or** $(B\subseteq \mathrm{N}))$* ***then***
$\mathrm{SAT}^{\mathrm{C}}(B)\in \mathbf{L}$
else if *$((B\subseteq \mathrm{E})$ **or** $(B\subseteq \mathrm{V}))$* ***then***
$\mathrm{SAT}^{\mathrm{C}}(B)$ *is $\leq_{\mathrm{m}}^{\log}$-complete for* $\mathbf{NL}$
else if *$(B\subseteq \mathrm{L})$* ***then***
$\mathrm{SAT}^{\mathrm{C}}(B)$ *is $\leq_{\mathrm{m}}^{\log}$-complete for* $\oplus\mathbf{L}$
else if *$(B\subseteq \mathrm{M})$* ***then***
$\mathrm{SAT}^{\mathrm{C}}(B)$ *is $\leq_{\mathrm{m}}^{\log}$-complete for* $\mathbf{P}$
else
$\mathrm{SAT}^{\mathrm{C}}(B)$ *is $\leq_{\mathrm{m}}^{\log}$-complete for* $\mathbf{NP}$.
There exists an algorithm which decides which of the cases above takes place.

Proof. 1. Let $B\subseteq \mathrm{R}_1$, and let C be a B-circuit. By Proposition 2.15 we obtain $f_C\in[B\cup\{id\}]\subseteq \mathrm{R}_1$. Hence $f_C(1,\dots 1)=1$, and $C\in \mathrm{SAT}^{\mathrm{C}}(B)$.

2. Let $B\subseteq D$, and let C be a B-circuit. By Proposition 2.15 we obtain $f_C\in[B\cup\{id\}]\subseteq \mathrm{D}$. Hence $f_C(0,\dots,0)=1$ or $f_C(1,\dots,1)=1$, and $C\in\mathrm{SAT}^{\mathrm{C}}(B)$.

3. If $B\subseteq \mathrm{N}=[\{\mathrm{non},1\}]$ then, by Proposition 2.19, we get $\mathrm{SAT}^{\mathrm{C}}(B)\leq_{\mathrm{m}}^{\log}\mathrm{SAT}^{\mathrm{C}}(\{\mathrm{non},1\})$. For a $\{\mathrm{non},1\}$-circuit $C(x_1,\dots,x_n)$, there exists an $a\in\{0,1\}^n$ such that $f_C(a)=1$ iff the "backward path" from the output-gate either ends in an input-gate or it ends in a constant 1-gate and has even length. This can be checked in logarithmic space.

4. Let $B\not\subseteq \mathrm{R}_1$, $B\not\subseteq \mathrm{N}$, and $B\subseteq \mathrm{V}$. An inspection of Figure 2.2 yields $[B]=\mathrm{V}_0$ or $[B]=\mathrm{V}$. Consequently $0\in \mathrm{V}_0\subseteq[B]\subseteq \mathrm{M}$, and by Proposition 3.6 we obtain $\mathrm{SAT}^{\mathrm{C}}(B)\equiv_{\mathrm{m}}^{\log}\mathrm{VAL}^{\mathrm{C}}(B)$. Now Theorem 3.4 yields that $\mathrm{SAT}^{\mathrm{C}}(B)$ is $\leq_{\mathrm{m}}^{\log}$-complete for $\mathbf{NL}$.

5. The case $B\not\subseteq \mathrm{R}_1$, $B\not\subseteq \mathrm{N}$, and $B\subseteq \mathrm{E}$ is treated in the same way; just replace V by E.

6. Let $B\not\subseteq \mathrm{R}_1$, $B\not\subseteq \mathrm{N}$, and $B\subseteq \mathrm{L}$. An inspection of Figure 2.2 yields $[B]=\mathrm{L}_0$ or $[B]=\mathrm{L}$. Consequently, $\mathrm{aut}\in \mathrm{L}_0\subseteq[B\cup\{\mathrm{id}\}]$ and, by Theorem 2.11, we have $B\subseteq \mathrm{L}=[\{\mathrm{aut},1\}]=[\{\mathrm{aut},1,\mathrm{id}\}]$. By Proposition 2.19 we obtain $\mathrm{SAT}^{\mathrm{C}}(\{\mathrm{aut}\})\leq_{\mathrm{m}}^{\log}\mathrm{SAT}^{\mathrm{C}}(B)\leq_{\mathrm{m}}^{\log}\mathrm{SAT}^{\mathrm{C}}(\{\mathrm{aut},1\})$.
Upper bound. For an $\{\mathrm{aut},1\}$-circuit $C(x_1,\dots,x_n)$ let m_0 be the number of paths from the output gate to a constant 1-gate, and for $i=1,\dots,n$, let m_i be the number of paths from the output gate to an x_i-gate. Obviously, x_i is fictive in f_C iff m_i is even. Consequently, C is satisfiable iff at least one of the numbers $m_0,m_1,\dots,m_n$ is odd. This can be checked by a $\mathbf{L}^{\oplus\mathbf{L}}$ computation. By Proposition 2.4

we have $\mathbf{L}^{\oplus\mathbf{L}} = \oplus\mathbf{L}$.
Lower Bound. We prove GOAP $\leq_{\mathrm{m}}^{\log}$ $\mathrm{SAT}^{\mathrm{C}}(\{\mathrm{aut}\})$ (cf. Theorem 2.5). Given a directed acyclic graph G whose vertices have outdegree 0 or 2, a start vertex s, and a target vertex t with outdegree 0. We construct in logarithmic space an $\{\mathrm{aut}\}$-circuit C with only one input variable x as follows: Every outdegree 2 vertex becomes an aut-gate, the vertex t becomes an x-gate, and every other outdegree 0 vertex becomes an aut-gate with both incoming edges from an x-gate. The vertex s becomes the output gate of C. If there is an odd number of paths in G from s to t, then $f_C = \mathrm{id}$, otherwise f_C is the unary constant 0 function.

7. Let $B \not\subseteq \mathrm{R}_1$, $B \not\subseteq \mathrm{D}$, $B \not\subseteq \mathrm{V}$, $B \not\subseteq \mathrm{E}$, and $B \subseteq \mathrm{M}$. An inspection of Figure 2.2 yields $0 \in \mathrm{S}_{11} \subseteq [B] \subseteq \mathrm{M}$. By Proposition 3.6 we obtain $\mathrm{SAT}^{\mathrm{C}}(B) \equiv_{\mathrm{m}}^{\log} \mathrm{VAL}^{\mathrm{C}}(B)$. Now Theorem 3.4 yields that $\mathrm{SAT}^{\mathrm{C}}(B)$ is $\leq_{\mathrm{m}}^{\log}$-complete for $\mathbf{P}$.

8. Let $B \not\subseteq \mathrm{R}_1$, $B \not\subseteq \mathrm{D}$, $B \not\subseteq \mathrm{L}$, and $B \not\subseteq \mathrm{M}$. An inspection of Figure 2.2 yields $\mathrm{S}_1 \subseteq [B]$. Observe $x \wedge \overline{y} \in \mathrm{S}_1$. By Theorem 3.5 we get that $\mathrm{SAT}^{\mathrm{C}}(B)$ is $\leq_{\mathrm{m}}^{\log}$-complete for $\mathbf{NP}$. □

Now we turn to the tautology problem. Remember that we defined (cf. Definition 2.16) for a B-circuit C its dual-circuit $\mathrm{dual}(C)$ be the $\mathrm{dual}(B)$-circuit which arises from C by replacing every f-gate by a $\mathrm{dual}(f)$-gate. Clearly, $f_{\mathrm{dual}(C)} = \mathrm{dual}(f_C)$. We will use this fact for the following duality principle.

Proposition 3.9. *For any finite set B of Boolean functions,* $\mathrm{TAUT}^{\mathrm{C}}(B) \equiv_{\mathrm{m}}^{\log} \overline{\mathrm{SAT}^{\mathrm{C}}(\mathrm{dual}(B))}$.

Proof. For a B-circuit C it holds that $C \in \mathrm{TAUT}^{\mathrm{C}}(B)$ iff $f_C \equiv 1$ iff $f_{\mathrm{dual}(C)} \equiv 0$ iff $\mathrm{dual}(C) \in \overline{\mathrm{SAT}^{\mathrm{C}}(\mathrm{dual}(B))}$. □

Using this proposition we are able to prove the main theorem on the complexity of the tautology problem for B-circuits ($\mathrm{TAUT}^{\mathrm{C}}(B)$).

Theorem 3.10. *Let be B a finite set of Boolean functions.*
if *(($B \subseteq \mathrm{R}_0$)* ***or*** *($B \subseteq \mathrm{D}$)* ***or*** *($B \subseteq \mathrm{N}$))* ***then***
$\mathrm{TAUT}^{\mathrm{C}}(B) \in \mathbf{L}$
else if *(($B \subseteq \mathrm{E}$)* ***or*** *($B \subseteq \mathrm{V}$))* ***then***
$\mathrm{TAUT}^{\mathrm{C}}(B)$ *is* $\leq_{\mathrm{m}}^{\log}$*-complete for* $\mathbf{NL}$
else if *($B \subseteq \mathrm{L}$)* ***then***
$\mathrm{TAUT}^{\mathrm{C}}(B)$ *is* $\leq_{\mathrm{m}}^{\log}$*-complete for* $\oplus\mathbf{L}$
else if *($B \subseteq \mathrm{M}$)* ***then***
$\mathrm{TAUT}^{\mathrm{C}}(B)$ *is* $\leq_{\mathrm{m}}^{\log}$*-complete for* $\mathbf{P}$
else
$\mathrm{TAUT}^{\mathrm{C}}(B)$ *is* $\leq_{\mathrm{m}}^{\log}$*-complete for* $\mathbf{coNP}$.
There exists an algorithm which decides which of the cases above takes place.

Proof. The theorem follows from Theorem 3.8 and Proposition 3.9 by using the facts $\mathrm{dual}(\mathrm{N}) = \mathrm{N}$, $\mathrm{dual}(\mathrm{D}) = \mathrm{D}$, $\mathrm{dual}(\mathrm{R}_0) = \mathrm{R}_1$, $\mathrm{dual}(\mathrm{V}) = \mathrm{E}$, $\mathrm{dual}(\mathrm{E}) = \mathrm{V}$, $\mathrm{dual}(\mathrm{L}) = \mathrm{L}$, and $\mathrm{dual}(\mathrm{M}) = \mathrm{M}$ (cf. Proposition 2.12.3) as well as $\mathbf{coL} = \mathbf{L}$,

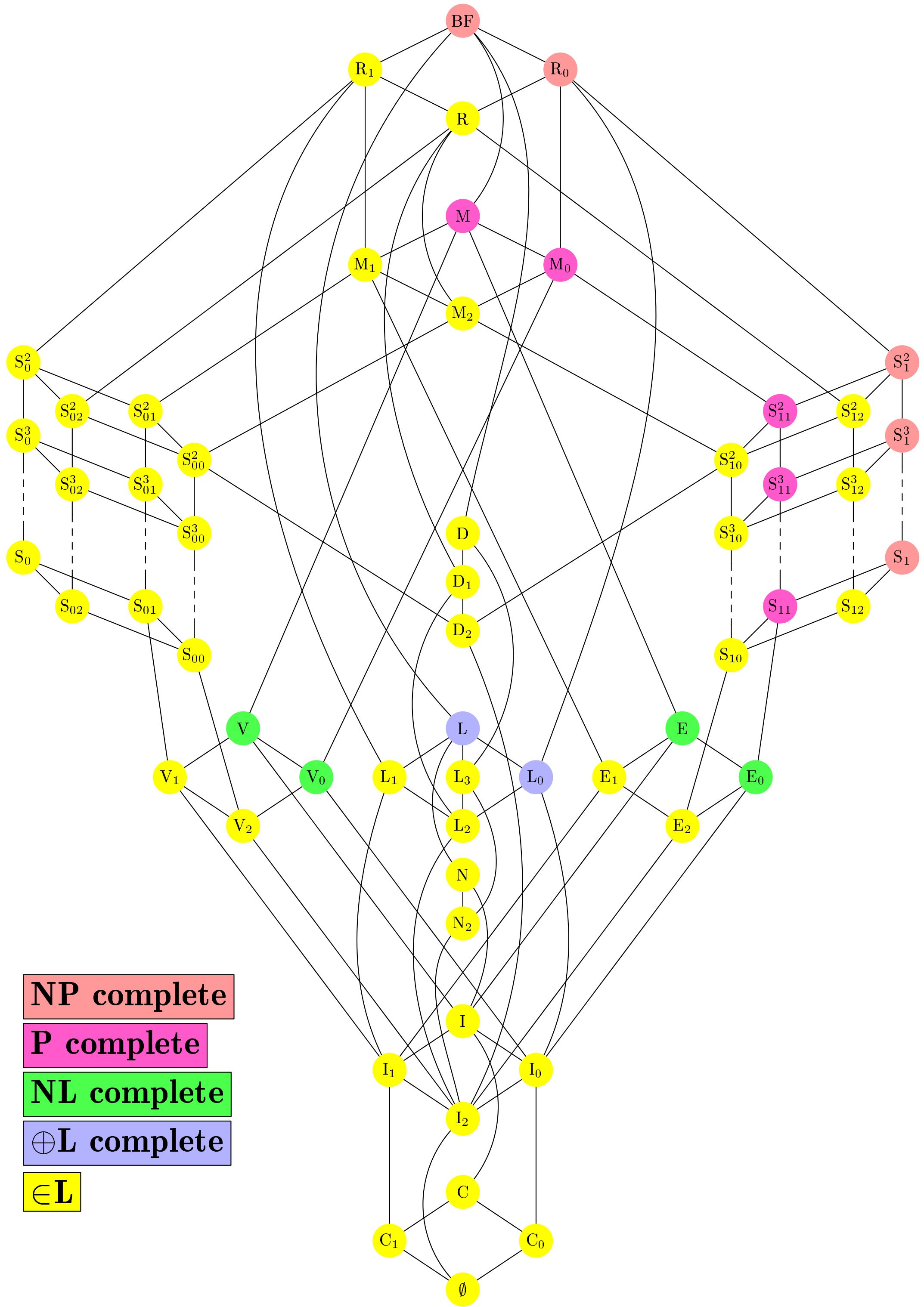


Fig. 3.4. The complexity of $\mathrm{SAT}^{\mathrm{C}}(B)$.

coNL = **NL**, **co⊕L** = **⊕L**, and **coP** = **P** (cf. Theorem 2.3). For example: If $B \not\subseteq \mathrm{R}_0$, $B \not\subseteq \mathrm{D}$, $B \not\subseteq \mathrm{N}$, and $B \subseteq \mathrm{V}$ then $\mathrm{dual}(B) \not\subseteq \mathrm{R}_1$, $\mathrm{dual}(B) \not\subseteq \mathrm{D}$, $\mathrm{dual}(B) \not\subseteq \mathrm{N}$, and $\mathrm{dual}(B) \subseteq \mathrm{E}$. By Theorem 3.8 the problem $\mathrm{SAT}^{\mathrm{C}}(\mathrm{dual}(B))$ is $\leq_{\mathrm{m}}^{\log}$-complete for **NL**, and by Proposition 3.9 the problem $\mathrm{TAUT}^{\mathrm{C}}(B)$ is $\leq_{\mathrm{m}}^{\log}$-complete for **coNL** = **NL**. □

3.4 Quantifiers

In this section we study the complexity of problems defined by B-circuits with quantified input variables. For $Q_1, Q_2, \ldots, Q_n \in \{\exists, \forall\}$ and $k \geq 1$, we call $Q_1 Q_2 \ldots Q_n$ a Σ_k*-string* (a Π_k*-string*, resp.) if $Q_1 = \exists$ ($Q_1 = \forall$, resp.) and there are at most $k-1$ alternations between $\exists$ and $\forall$ in $Q_1 Q_2 \ldots Q_n$. For a finite set B of Boolean functions define:

PROBLEM: $\Sigma_k^{\mathrm{C}}(B)$
INSTANCE: A B-circuit $C(x_1, \ldots, x_n)$ and a Σ_k-string $Q_1 \ldots Q_n$
QUESTION: $Q_1 x_1 \ldots Q_n x_n \, f_C(x_1, \ldots, x_n) = 1$?

PROBLEM: $\Pi_k^{\mathrm{C}}(B)$
INSTANCE: A B-circuit $C(x_1, \ldots, x_n)$ and a Π_k-string $Q_1 \ldots Q_n$
QUESTION: $Q_1 x_1 \ldots Q_n x_n \, f_C(x_1, \ldots, x_n) = 1$?

PROBLEM: $\mathrm{QBF}^{\mathrm{C}}(B)$
INSTANCE: A B-circuit $C(x_1, \ldots, x_n)$ and quantifiers $Q_1, \ldots, Q_n \in \{\exists, \forall\}$
QUESTION: $Q_1 x_1 \ldots Q_n x_n \, f_C(x_1, \ldots, x_n) = 1$?

Notice that $\Sigma_1^{\mathrm{C}}(B) = \mathrm{SAT}^{\mathrm{C}}(B)$ and $\Pi_1^{\mathrm{C}}(B) = \mathrm{TAUT}^{\mathrm{C}}(B)$ have already been treated in the previous section. Here we concentrate on the case $k \geq 2$.

It is obvious that $\Sigma_k^{\mathrm{C}}(B) \in \boldsymbol{\Sigma_k^{\mathrm{p}}}$, $\Pi_k^{\mathrm{C}}(B) \in \boldsymbol{\Pi_k^{\mathrm{p}}}$, and $\mathrm{QBF}^{\mathrm{C}}(B) \in$ **PSPACE** for every finite set B of Boolean functions. The following results can easily be derived from the literature, where these results are proved for the special case of B-formulas.

Theorem 3.11. *[SM73] Let B be a finite set of Boolean functions such that $[B] = \mathrm{BF}$, and let $k \geq 1$. Then $\Sigma_k^{\mathrm{C}}(B)$ is $\leq_{\mathrm{m}}^{\log}$-complete for $\boldsymbol{\Sigma_k^{\mathrm{p}}}$, $\Pi_k^{\mathrm{C}}(B)$ is $\leq_{\mathrm{m}}^{\log}$-complete for $\boldsymbol{\Pi_k^{\mathrm{p}}}$, and $\mathrm{QBF}^{\mathrm{C}}(B)$ is $\leq_{\mathrm{m}}^{\log}$-complete for* **PSPACE**.

In the following we show that for sets B of monotone Boolean functions the complexity of $\Sigma_k^{\mathrm{C}}(B)$, $\Pi_k^{\mathrm{C}}(B)$ and $\mathrm{QBF}^{\mathrm{C}}(B)$ can be related to the complexity of $\mathrm{VAL}^{\mathrm{C}}(B)$.

Proposition 3.12. *Let $B \subseteq \mathrm{M}$ be a finite set of Boolean functions. Then $\Sigma_k^{\mathrm{C}}(B) \equiv_{\mathrm{m}}^{\log} \Pi_k^{\mathrm{C}}(B) \equiv_{\mathrm{m}}^{\log} \mathrm{QBF}^{\mathrm{C}}(B) \equiv_{\mathrm{m}}^{\log} \mathrm{VAL}^{\mathrm{C}}(B)$ for every $k \geq 2$.*

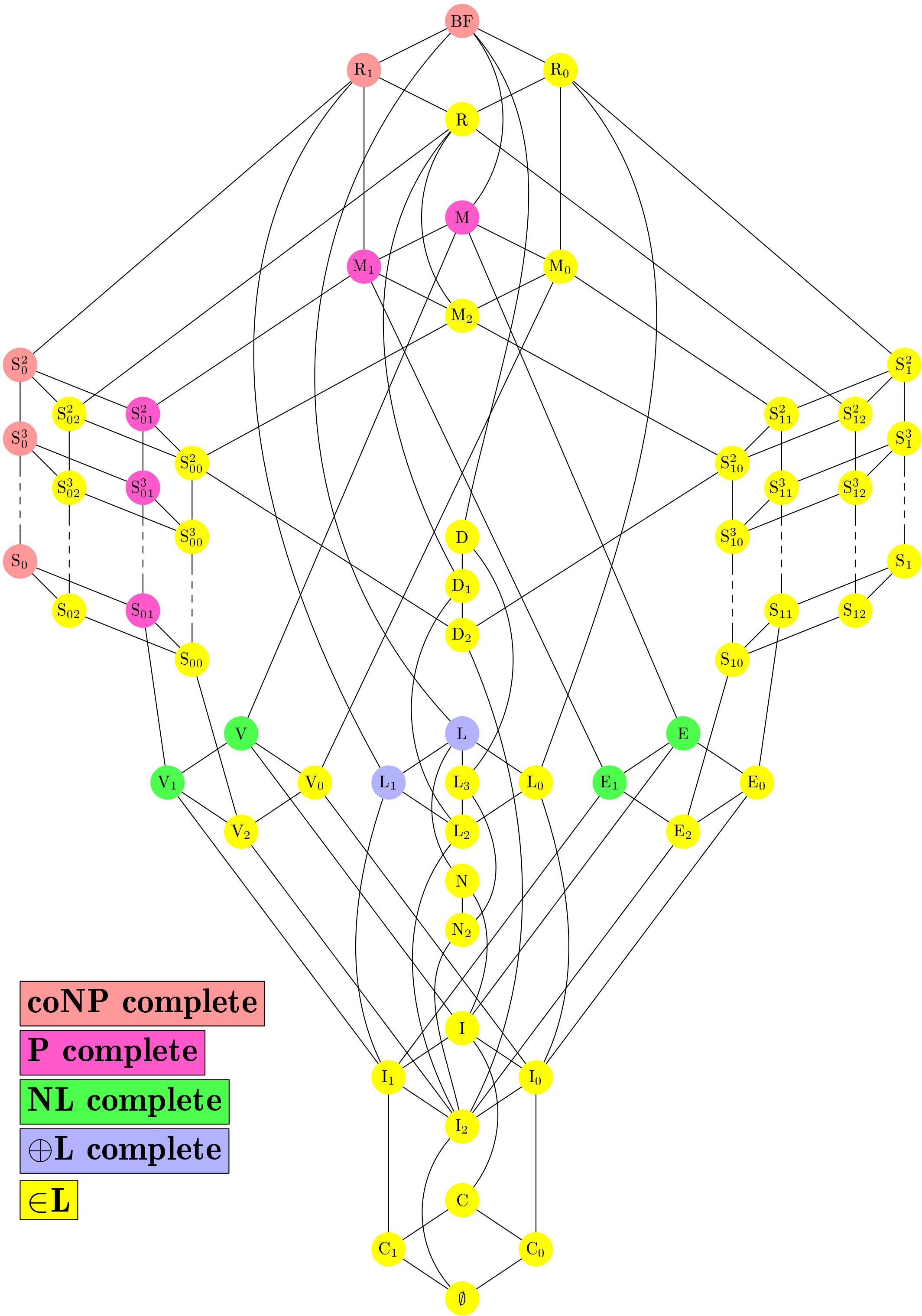


Fig. 3.5. The complexity of $\mathrm{TAUT}^{\mathrm{C}}(B)$.

Proof. First we show that $\Sigma_k^{\mathrm{C}}(B) \leq_{\mathrm{m}}^{\log} \mathrm{VAL}^{\mathrm{C}}(B)$, $\Pi_k^{\mathrm{C}}(B) \leq_{\mathrm{m}}^{\log} \mathrm{VAL}^{\mathrm{C}}(B)$, and $\mathrm{QBF}^{\mathrm{C}}(B) \leq_{\mathrm{m}}^{\log} \mathrm{VAL}^{\mathrm{C}}(B)$. Let $C(x_1, \ldots, x_n)$ be a B-circuit and let $Q_1, \ldots, Q_n \in \{\exists, \forall\}$. Defining $a_i = 1$ iff $Q_i = \exists$ for $1 \leq i \leq n$ we obtain $Q_1x_1Q_2x_2\ldots Q_nx_nf_C(x_1, \ldots, x_n) = 1$ iff $f(a_1, \ldots, a_n) = 1$.

Next we show $\mathrm{VAL}^{\mathrm{C}}(B) \leq_{\mathrm{m}}^{\log} \Sigma_k^{\mathrm{C}}(B)$, $\mathrm{VAL}^{\mathrm{C}}(B) \leq_{\mathrm{m}}^{\log} \Pi_k^{\mathrm{C}}(B)$, and $\mathrm{VAL}^{\mathrm{C}}(B) \leq_{\mathrm{m}}^{\log} \mathrm{QBF}^{\mathrm{C}}(B)$. Convert a B-circuit $C(x_1, \ldots, x_n)$ with an input string $a \in \{0,1\}^n$ into a B-circuit C' with two input-variables x and z as follows: Every input-gate with value 0 becomes a z-gate, and every input gate with value 1 becomes an x-gate. Finally note that $(C, a) \in \mathrm{VAL}^{\mathrm{C}}(B)$ iff $\exists x \forall z C' \in \Sigma_k^{\mathrm{C}}(B)$ iff $\forall z \exists x C' \in \Pi_k^{\mathrm{C}}(B)$ iff $\forall z \exists x C' \in \mathrm{QBF}^{\mathrm{C}}(B)$, which can be checked by expanding $\exists x \forall z C'$ and $\forall z \exists x C'$ into a quantor-free representation. □

Theorem 3.13. *Let be B a finite set of Boolean functions, and let $k \geq 2$.*

if *($B \subseteq \mathrm{N}$)* ***then***
$\Sigma_k^{\mathrm{C}}(B), \Pi_k^{\mathrm{C}}(B), \mathrm{QBF}^{\mathrm{C}}(B) \in \mathbf{L}$
else if *(($B \subseteq \mathrm{E}$)* ***or*** *($B \subseteq \mathrm{V}$))* ***then***
$\Sigma_k^{\mathrm{C}}(B), \Pi_k^{\mathrm{C}}(B)$*, and* $\mathrm{QBF}^{\mathrm{C}}(B)$ *are complete for* $\mathbf{NL}$
else if *($B \subseteq \mathrm{L}$)* ***then***
$\Sigma_k^{\mathrm{C}}(B), \Pi_k^{\mathrm{C}}(B)$*, and* $\mathrm{QBF}^{\mathrm{C}}(B)$ *are complete for* $\oplus\mathbf{L}$
else if *($B \subseteq \mathrm{M}$)* ***then***
$\Sigma_k^{\mathrm{C}}(B), \Pi_k^{\mathrm{C}}(B)$*, and* $\mathrm{QBF}^{\mathrm{C}}(B)$ *are complete for* $\mathbf{P}$
else
$\Sigma_k^{\mathrm{C}}(B)$ *is* $\leq_{\mathrm{m}}^{\log}$*-complete for* $\boldsymbol{\Sigma}_{\boldsymbol{k}}^{\mathbf{P}}$
$\Pi_k^{\mathrm{C}}(B)$ *is* $\leq_{\mathrm{m}}^{\log}$*-complete for* $\boldsymbol{\Pi}_{\boldsymbol{k}}^{\mathbf{P}}$
$\mathrm{QBF}^{\mathrm{C}}(B)$ *is* $\leq_{\mathrm{m}}^{\log}$*-complete for* $\mathbf{PSPACE}$.

There exists an algorithm which decides which of the cases above takes place.

Proof. 1. If $B \subseteq \mathrm{N}$ then, by Proposition 2.19 and Theorem 2.11, we get $\Sigma_k^{\mathrm{C}}(B) \leq_{\mathrm{m}}^{\log} \Sigma_k^{\mathrm{C}}(\{\mathrm{non}, 1\})$, $\Pi_k^{\mathrm{C}}(B) \leq_{\mathrm{m}}^{\log} \Pi_k^{\mathrm{C}}(\{\mathrm{non}, 1\})$, and $\mathrm{QBF}^{\mathrm{C}}(B) \leq_{\mathrm{m}}^{\log} \mathrm{QBF}^{\mathrm{C}}(\{\mathrm{non}, 1\})$. For a $\{\mathrm{non}, 1\}$-circuit C we have $Q_1x_1Q_2x_2\ldots Q_nx_nf_C(x_1, \ldots, x_n) = 1$ iff the "backward path" from the output gate either ends in an input gate with an existentially quantified variable or it ends in a 1-gate and has even length. This can be checked in logarithmic space.

2. Let $B \not\subseteq \mathrm{N}$ and ($B \subseteq \mathrm{V}$ or $B \subseteq \mathrm{E}$). By Proposition 3.12 we have $\Sigma_k^{\mathrm{C}}(B) \equiv_{\mathrm{m}}^{\log} \Pi_k^{\mathrm{C}}(B) \equiv_{\mathrm{m}}^{\log} \mathrm{QBF}^{\mathrm{C}}(B) \equiv_{\mathrm{m}}^{\log} \mathrm{VAL}^{\mathrm{C}}(B)$, and by Theorem 3.4 we obtain that these problems are $\leq_{\mathrm{m}}^{\log}$-complete for $\mathbf{NL}$.

3. Let $B \not\subseteq \mathrm{N}$ and $B \subseteq \mathrm{L}$. An inspection of Figure 2.2 yields $\mathrm{L}_2 \subseteq [B] \subseteq \mathrm{L}$. Define $h(x, y, z) =_{\mathrm{def}} x \oplus y \oplus z$. By Theorem 2.11 we get $h \in \mathrm{L}_2$ and $\mathrm{L} = [\{\mathrm{aut}, 1\}]$, and by Proposition 2.19 we obtain $\Sigma_k^{\mathrm{C}}(\{h\}) \leq_{\mathrm{m}}^{\log} \Sigma_k^{\mathrm{C}}(B) \leq_{\mathrm{m}}^{\log} \Sigma_k^{\mathrm{C}}(\{\mathrm{aut}, 1\})$, $\Pi_k^{\mathrm{C}}(\{h\}) \leq_{\mathrm{m}}^{\log} \Pi_k^{\mathrm{C}}(B) \leq_{\mathrm{m}}^{\log} \Pi_k^{\mathrm{C}}(\{\mathrm{aut}, 1\})$, and $\mathrm{QBF}^{\mathrm{C}}(\{h\}) \leq_{\mathrm{m}}^{\log} \mathrm{QBF}^{\mathrm{C}}(B) \leq_{\mathrm{m}}^{\log} \mathrm{QBF}^{\mathrm{C}}(\{\mathrm{aut}, 1\})$.

Upper bound. For an $\{\mathrm{aut}, 1\}$-circuit $C(x_1, \ldots, x_n)$ let m_0 be the number of paths from the output gate to a constant 1-gate, and for $i = 1, \ldots, n$, let m_i be the number of paths from the output gate to an x_i-gate. Obviously, x_i is fictive in f_C iff m_i is even. We conclude:

$Q_1x_1 \ldots Q_nx_nf_C(x_1,\ldots,x_n) = 1$ iff
iff either $f_C(0^n) = 1$ and $x_1,\ldots,x_n$ are fictive or there exists a non-fictive x_i such that $Q_i = \exists$ and either $i = n$ or $x_{i+1},\ldots,x_n$ are fictive.
iff either m_0 is odd and $m_1,\ldots,m_n$ are even or there exists an i such that $Q_i = \exists$, m_i is odd and either $i = n$ or $m_{i+1},\ldots,m_n$ are even.

The latter condition can be checked by a $\mathbf{L}^{\oplus\mathbf{L}}$ computation. By Theorem 2.4 we have $\mathbf{L}^{\oplus\mathbf{L}} = \oplus\mathbf{L}$.
Lower Bound. By Theorem 2.5 it is sufficient to prove GOAP $\leq_{\mathrm{m}}^{\log} \Pi_2^{\mathrm{C}}(\{h\})$ and $\overline{\text{GOAP}} \leq_{\mathrm{m}}^{\log} \Sigma_2^{\mathrm{C}}(\{h\})$. Given a directed acyclic graph G whose vertices have outdegree 0 or 2, a start vertex s, and a target vertex t with outdegree 0. We construct in logarithmic space a $\{h\}$-circuit C with input variables x, y and z as follows: Every outdegree 2 vertex becomes a h-gate where the additional incoming edge comes from a z-gate. The vertex t becomes an x-gate, and every other outdegree 0 vertex becomes a z-gate. It is not hard to see that $f_C(x,z) = g(z)$ for some Boolean function g if the number of paths in G from s to t is even and $f_C(x,z) = x \oplus g(z)$ otherwise. Construct a $\{h\}$-circuit C' such that $f_{C'}(x,y,z) = f_C(x,z) \oplus f_C(y,z) \oplus z$ (note that in the circuit which represents $f_C(y,z)$ the original target-gate is labeled with y). Now, if the number of paths in G from s to t is even then $f_{C'}(x,y,z) = z$ and consequently $\exists z\forall x\forall yC' \in \Sigma_2^{\mathrm{C}}(\{h\})$ and $\forall z\exists x\exists yC' \notin \Pi_2^{\mathrm{C}}(\{h\})$. On the other hand, if the number of paths in G from s to t is odd then $f_{C'}(x,y,z) = x \oplus y \oplus z$ and consequently $\forall z\exists x\exists yC' \in \Pi_2^{\mathrm{C}}(\{h\})$ and $\exists z\forall x\forall yC' \notin \Sigma_2^{\mathrm{C}}(\{h\})$.

4. Let $B \not\subseteq \mathrm{V}$, $B \not\subseteq \mathrm{E}$, and $B \subseteq \mathrm{M}$. By Proposition 3.12 we have $\Sigma_k^{\mathrm{C}}(B) \equiv_{\mathrm{m}}^{\log} \Pi_k^{\mathrm{C}}(B) \equiv_{\mathrm{m}}^{\log} \mathrm{QBF}^{\mathrm{C}}(B) \equiv_{\mathrm{m}}^{\log} \mathrm{VAL}^{\mathrm{C}}(B)$, and by Theorem 3.4 we obtain that these problems are $\leq_{\mathrm{m}}^{\log}$-complete for $\mathbf{P}$.

5. Let $B \not\subseteq \mathrm{M}$ and $B \not\subseteq \mathrm{L}$. An inspection of Figure 2.2 yields $\mathrm{S}_{02} \subseteq [B]$, $\mathrm{S}_{12} \subseteq [B]$, or $\mathrm{D}_1 \subseteq [B]$. Define the functions $h_1(x,u,v,z) =_{\mathrm{def}} (x \vee u) \cdot \overline{v} \cdot z \vee u$, $h_2(x,u,v,z) =_{\mathrm{def}} (x \vee u) \cdot \overline{v} \cdot z \vee u \cdot z$, and $h_3(x,u,v,z) =_{\mathrm{def}} (x \vee u) \cdot \overline{v} \cdot z \vee u \cdot (x \vee \overline{v} \vee z)$. Observe that h_1, h_2, and h_3 are 0-reproducing and 1-reproducing, that h_1 is 0-separating (note that for any $(x,y,v,z) \in h_1^{-1}(0)$ $u = 0$ holds), that h_2 is 1-separating (note that for any $(x,y,v,z) \in h_2^{-1}(1)$ $z = 1$ holds), and that h_3 is self-dual (this can be checked by inspecting the truthtable). Hence $h_1 \in \mathrm{R}_0 \cap \mathrm{R}_1 \cap \mathrm{S}_0 = \mathrm{S}_{02}$, $h_2 \in \mathrm{R}_0 \cap \mathrm{R}_1 \cap \mathrm{S}_1 = \mathrm{S}_{12}$, and $h_3 \in \mathrm{R}_0 \cap \mathrm{R}_1 \cap \mathrm{D} = \mathrm{D}_1$. Thus there exists an $h \in [B]$ such that $h(x,u,0,1) = x \vee u$ and $h(x,u,v,z) \leq u$ for all $(v,z) \neq (0,1)$. Observe that $B \cup \{0,1\}$ is complete. By Theorem 3.11 it is sufficient to prove $\Sigma_k^{\mathrm{C}}(B \cup \{0,1\}) \leq_{\mathrm{m}}^{\log} \Sigma_k^{\mathrm{C}}(B)$ and $\Pi_k^{\mathrm{C}}(B \cup \{0,1\}) \leq_{\mathrm{m}}^{\log} \Pi_k^{\mathrm{C}}(B)$ for $k \geq 2$ as well as $\mathrm{QBF}^{\mathrm{C}}(B \cup \{0,1\}) \leq_{\mathrm{m}}^{\log} \mathrm{QBF}^{\mathrm{C}}(B)$, since the corresponding upper bounds clearly hold. For a $(B \cup \{0,1\})$-circuit C we construct a B-circuit C_1 with two new input variables such that $f_C(W,X,Y) = f_{C_1}(W,X,Y,0,1)$ where W,X,Y stand for sets of variables. Now we construct a B-circuit C_2 such that $f_{C_2}(W,X,Y,u,v,z) = h(f_{C_1}(W,X,Y,v,z),u,v,z)$. Hence we have $f_{C_2}(W,X,Y,u,0,1) = f_C(W,X,Y) \vee u$ and $f_{C_2}(W,X,Y,u,v,z) \leq u$ for $(v,z) \neq (0,1)$. We observe that for all w,

$$\begin{aligned}\exists X \forall Y f_C(W,X,Y) &\text{ iff } \exists X \forall Y \forall u (f_C(W,X,Y) \vee u)\\ &\text{ iff } \exists v \exists z \exists X \forall Y \forall u f_{C_2}(W,X,Y,u,v,z)\end{aligned}$$

and

$$\begin{aligned}\forall X \exists Y f_C(W,X,Y) &\text{ iff } \forall u \forall X \exists Y (f_C(W,X,Y) \vee u)\\ &\text{ iff } \forall u \forall X \exists Y \exists v \exists z f_{C_2}(W,X,Y,u,v,z).\end{aligned}$$

Since the set W can contain arbitrarily quantified variables this gives the desired reductions. □

3.5 Counting functions

In this section we study the complexity of functions which count the number of satisfying inputs of a B-circuit. For any finite set B of Boolean functions define the *counting function* as follows:

PROBLEM:	$\#^{\mathrm{C}}(B)$
INSTANCE:	A B-circuit C
OUTPUT:	$\#_1(C)$

It is obvious that $\#^{\mathrm{C}}(B) \in \#\mathbf{P}$ for every finite set B of Boolean functions. The following results on the complexity of $\#^{\mathrm{C}}(B)$ can be found in or easily be derived from the literature.

Theorem 3.14 ([Sim75]). *Let B be a finite set of Boolean functions.*

1. *If $[B] = \mathrm{BF}$ then $\#^{\mathrm{C}}(B)$ is $\leq_{\mathrm{m}}^{\log}$-complete for $\#\mathbf{P}$.*
2. *$\#^{\mathrm{C}}(\{\mathrm{et}, \mathrm{vel}, \mathrm{non}\})$ restricted to circuits in conjunctive normal form is $\leq_{\mathrm{m}}^{\log}$-complete for $\#\mathbf{P}$.*

Other examples for $\#\mathbf{P}$ complete functions can be found in [Pap94], pp. 442f.

To study the complexity of $\#^{\mathrm{C}}(B)$ in the general case we will use the following propositions which are analogues to Proposition 2.19 and Proposition 3.7 and which are proved in the same way.

Proposition 3.15. *Let B and B' be finite sets of Boolean functions.*

1. *If $B \subseteq \langle B' \rangle$ then $\#^{\mathrm{C}}(B) \leq_{\mathrm{m}}^{\log} \#^{\mathrm{C}}(B')$.*
2. *If $\langle B \rangle = \langle B' \rangle$ then $\#^{\mathrm{C}}(B) \equiv_{\mathrm{m}}^{\log} \#^{\mathrm{C}}(B')$.*

Proposition 3.16. *If B is a finite set of Boolean functions containing* et *then $\#^{\mathrm{C}}(B \cup \{1\}) \leq_{\mathrm{m}}^{\log} \#^{\mathrm{C}}(B)$.*

Proof. Note that the reduction in proof of Proposition 3.7 is *parsimonious.* □

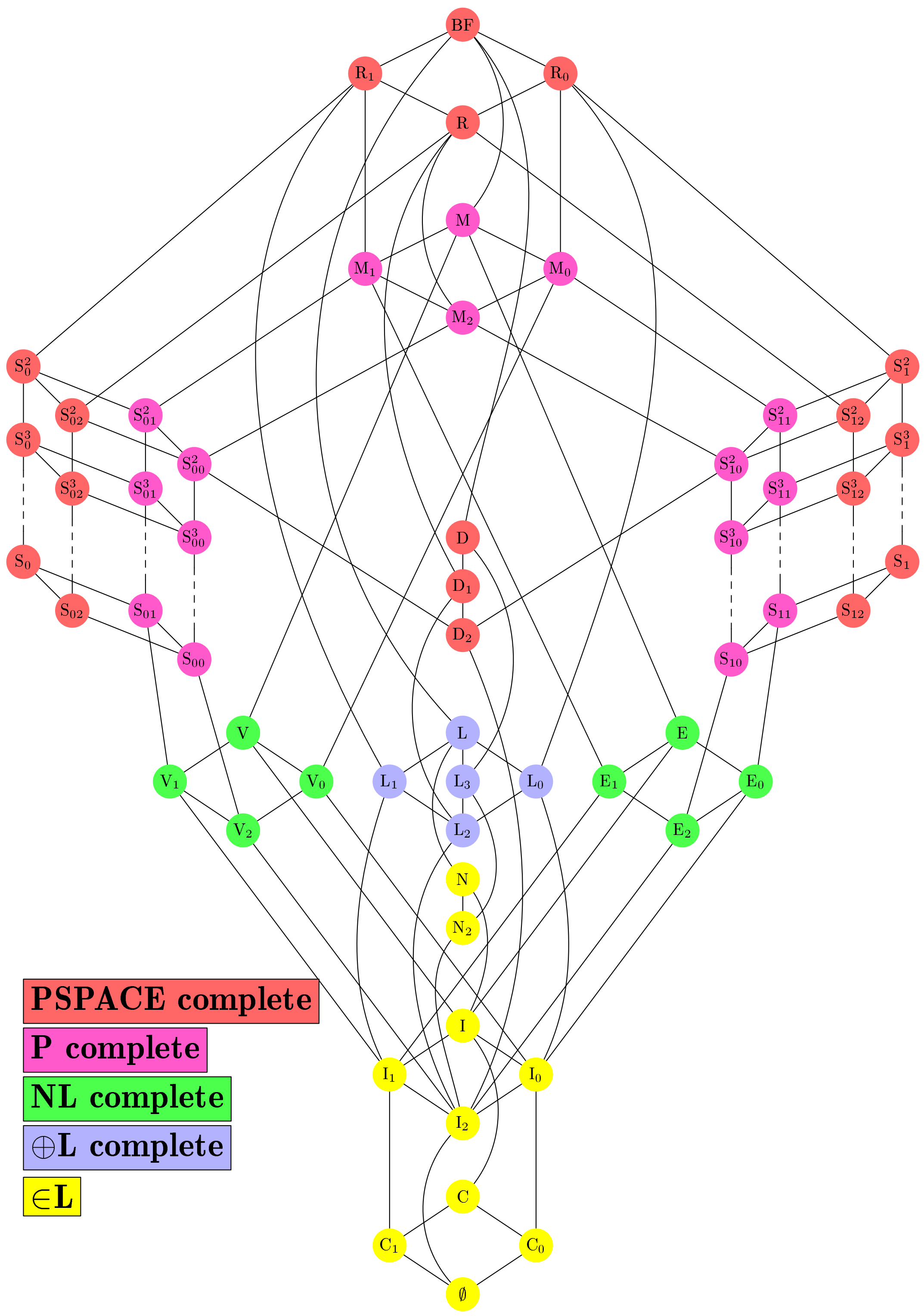


Fig. 3.6. The complexity of $\mathrm{QBF}^{\mathrm{C}}(B)$.

We will use the following obvious duality principle.

Proposition 3.17.
Let B be a finite set of Boolean functions, then $\#^{\mathrm{C}}(B) \leq^{log}_{\text{1-T}} \#^{\mathrm{C}}(\mathrm{dual}(B))$ and $\#^{\mathrm{C}}(\mathrm{dual}(B)) \leq^{log}_{\text{1-T}} \#^{\mathrm{C}}(B)$.

Proof. This is a direct consequence of Proposition 2.18. □

Next we prove a lemma about the representation of $\#\mathbf{P}$ functions by $\#^{\mathrm{C}}(B)$ functions.

Lemma 3.18. *Let B be a finite set of Boolean functions.*

1. *If $[B] \supseteq \mathrm{S}_1$ then $\#^{\mathrm{C}}(B)$ is $\leq^{\log}_{\mathrm{m}}$-complete for $\#\mathbf{P}$, i.e., for every function $f \in \#\mathbf{P}$ there exists a logspace computable function h generating B-circuits such that $f(x) = \#_1(h(x))$.*
2. *If $[B] \supseteq \mathrm{S}_{10}$ or $[B] \supseteq \mathrm{S}_{00}$ then $\#^{\mathrm{C}}(B)$ is $\leq^{log}_{\text{1-T}}$-complete for $\#\mathbf{P}$. More specifically, for every function $f \in \#\mathbf{P}$ there exist logspace computable functions h_1, h_2 generating B-circuits and logspace computable functions h'_1, h'_2 such that $f(x) = \#_1(h_1(x)) - h'_1(x) = h'_2(x) - \#_1(h_2(x))$.*

Proof. 1. Because of $\mathrm{et} \in \mathrm{S}_1$ and by Proposition 3.16 we obtain $\#^{\mathrm{C}}(B \cup \{1\}) \leq^{\log}_{\mathrm{m}} \#^{\mathrm{C}}(B)$. Observe $[B \cup \{1\}] = \mathrm{BF}$. Now Theorem 3.14 yields that $\#^{\mathrm{C}}(B)$ is $\leq^{\log}_{\mathrm{m}}$-complete for $\#\mathbf{P}$.

2. Let $[B] \supseteq \mathrm{S}_{10}$. Because of $\mathrm{et} \in \mathrm{S}_{10}$ and Proposition 3.16 we obtain $\#^{\mathrm{C}}(B \cup \{1\}) \leq^{\log}_{\mathrm{m}} \#^{\mathrm{C}}(B)$. By Figure 2.2 and Proposition 2.11 we have $\{\mathrm{et}, \mathrm{vel}\} \subseteq \mathrm{M}_1 = [\mathrm{S}_{10} \cup \{1\}] \subseteq [B \cup \{1\}]$. By Proposition 3.15.1 it is sufficient to prove the statement for $B = \{\mathrm{et}, \mathrm{vel}\}$, and by Theorem 3.14.2 it is sufficient to prove $\#^{\mathrm{C}}(\{\mathrm{et}, \mathrm{vel}, \mathrm{non}\})(C) = \#^{\mathrm{C}}(\{\mathrm{et}, \mathrm{vel}\})(h_1(C)) - h'_1(C) = h'_2(C) - \#^{\mathrm{C}}(\{\mathrm{et}, \mathrm{vel}\})(h_2(C))$ for suitable logspace computable functions h_1, h'_1, h_2, h'_2 and all circuits C in conjunctive normal form. Let $C(x_1, \ldots, x_n)$ be such a circuit. Choose new variables $y_1, \ldots, y_n$, and let $C_1(x_1, \ldots, x_n, y_1, \ldots, y_n)$ be that circuit which arises from C when every $\overline{x}_i$ is replaced with y_i. By defining $C_2 =_{\mathrm{def}} \bigwedge_{i=1}^{n} (\overline{x_i} \leftrightarrow y_i) \wedge C_1$ we obtain $C_2 \equiv \overline{C_3} \wedge C_4$ where $C_3 =_{\mathrm{def}} \bigvee_{i=1}^{n} (x_i \wedge y_i)$ and $C_4 =_{\mathrm{def}} \bigwedge_{i=1}^{n} (x_i \vee y_i) \wedge C_1$. We conclude $\#_1(C) = \#_1(C_2) = \#_1(\overline{C_3} \wedge C_4) = \#_1(C_3 \vee C_4) - \#_1(C_3) = \#_1(C_3 \vee C_4) - (4^n - 3^n)$.
On the other hand, by Theorem 2.3 we know that for every function $f \in \#\mathbf{P}$ there exist a function $f' \in \#\mathbf{P}$ and a logspace computable function g such that $f(x) = g(x) - f'(x)$ for all x. By the above we get a logspace computable function h, which generates B-circuits and a logspace computable function g' such that $f'(x) = \#_1(h(x)) - g'(x)$ for all x. Hence $f(x) = (g(x) + g'(x)) - \#_1(h(x))$ for all x.
Now let $[B] \supseteq \mathrm{S}_{00}$. By Proposition 2.12 we obtain $[\mathrm{dual}(B)] = \mathrm{dual}([B]) \supseteq \mathrm{dual}(\mathrm{S}_{00}) = \mathrm{S}_{10}$. By the S_{10}-case we get for every $f \in \#\mathbf{P}$ logspace computable functions h'_1, h'_2, h_1, h_2 such that $f(x) = \#_1(h_1(x)) - h'_1(x) = h'_2(x) - \#_1(h_2(x))$ where $h_1(x)$ and $h_2(x)$ are $\mathrm{dual}(B)$-circuits. Let n_1 (n_2, resp.) be the number of variables associated with $h_1(x)$ (h_2, resp.). By Proposition 2.18 we get

$f(x) = (2^{n_1} - \#_1(\mathrm{dual}(h_1(x))) - h_1'(x) = (2^{n_1} - h_1'(x)) - \#_1(\mathrm{dual}(h_1(x)))$ and $f(x) = h_2'(x) - (2^{n_2} - \#_1(\mathrm{dual}(h_2(x))) = \#_1(\mathrm{dual}(h_2(x))) - (2^{n_2} - h_2'(x))$ where $\mathrm{dual}(h_1(x))$ and $\mathrm{dual}(h_2(x))$ are B-circuits. □

Now we are able to prove our main result on the complexity of $\#^{\mathrm{C}}(B)$.

Theorem 3.19. *Let be B a finite set of Boolean functions.*

if *(($B \subseteq \mathrm{N}$)* ***or*** *($B \subseteq \mathrm{D}$))* ***then***
$\#^{\mathrm{C}}(B) \in \mathbf{FL}$
else if *(($B \subseteq \mathrm{E}$)* ***or*** *($B \subseteq \mathrm{V}$))* ***then***
$\#^{\mathrm{C}}(B)$ *is* $\leq_{1\text{-T}}^{log}$ *-complete for* $\mathrm{FL}_{\|}^{\mathbf{NL}}[O(\log n)]$
else if *($B \subseteq \mathrm{L}$)* ***then***
$\#^{\mathrm{C}}(B)$ *is* $\leq_{1\text{-T}}^{log}$ *-hard for* $\mathrm{FL}_{\|}^{\oplus\mathbf{L}}[1]$ *and*
$\#^{\mathrm{C}}(B) \in \mathrm{FL}_{\|}^{\oplus\mathbf{L}}[2]$
else if *($B \subseteq \mathrm{R}_1$)* ***then***
$\#^{\mathrm{C}}(B)$ *is* $\leq_{1\text{-T}}^{log}$ *-complete for* $\#\mathbf{P}$,
but not $\leq_{\mathrm{m}}^{\log}$*-complete for* $\#\mathbf{P}$
else if *($B \subseteq \mathrm{M}$)* ***then***
$\#^{\mathrm{C}}(B)$ *is* $\leq_{1\text{-T}}^{log}$ *-complete for* $\#\mathbf{P}$.
If $\#^{\mathrm{C}}(B)$ *is* $\leq_{\mathrm{m}}^{\log}$*-complete for* $\#\mathbf{P}$ *then* $\mathbf{P} = \mathbf{NP}$.
else
$\#^{\mathrm{C}}(B)$ *is* $\leq_{\mathrm{m}}^{\log}$*-complete for* $\#\mathbf{P}$.

There exists an algorithm which decides which of the cases above takes place.

Proof. 1. If $B \subseteq \mathrm{N}$ then, by Theorem 2.11, we obtain $B \subseteq [\{\mathrm{non}, 1\}]$ and hence $\#^{\mathrm{C}}(B) \leq_{\mathrm{m}}^{\log} \#^{\mathrm{C}}(\{\mathrm{non}, 1\})$. Let C be a $\{\mathrm{non}, 1\}$-circuit with n input variables. Follow the "backward path" from the output gate as long as an indegree 0 gate is reached. If this is a 1-gate and the path has even length then $\#_1(C) = 2^n$. If this is a 1-gate and the path has odd length then $\#_1(C) = 0$. If this is an input-gate then $\#_1(C) = 2^{n-1}$. This algorithm works in logspace.

2. If $B \subseteq \mathrm{D}$ and $C(x_1, \ldots, x_n)$ is a B-circuit then $\#_1(C) = 2^{n-1}$.

3. Let $B \not\subseteq \mathrm{N}$ and $B \subseteq \mathrm{E}$. By Figure 2.2 and Theorem 2.11 we obtain $[\{\mathrm{et}\}] = \mathrm{E}_2 \subseteq [B] \subseteq \mathrm{E} = [\{\mathrm{et}, 0, 1\}]$. By Proposition 3.15 we obtain $\#^{\mathrm{C}}(\{\mathrm{et}\}) \leq_{\mathrm{m}}^{\log} \#^{\mathrm{C}}(B) \leq_{\mathrm{m}}^{\log} \#^{\mathrm{C}}(\{\mathrm{et}, 0, 1\})$.
Upper bound. For a non-constant $\{\mathrm{et}, 0, 1\}$-circuit $C(x_1, \ldots, x_n)$ we have $f_C(x_1, \ldots, x_n) = \bigwedge_{i \in I_C} x_i$, where $I_C =_{\mathrm{def}} \{i \mid x_i \text{ is non-fictive in } f_C\}$, and hence $\#_1(C) = 2^{n-|I_C|}$. It is sufficient to prove that the function $C \to |I_C|$ is in $\mathbf{FL}_{\|}^{\mathbf{NL}}[O(\log n)]$, since we can check whether C represents a constant function by two $\mathbf{NL}$-queries (cf. Theorem 3.8 and Theorem 3.10). Observe that x_i is non-fictive iff $f_C(1^n) = 1$ and $f_C(1^{i-1}01^{n-i}) = 0$. By Theorem 3.4 the problem of evaluating C is in $\mathbf{NL}$ and hence the set $S_C =_{\mathrm{def}} \{(C, i) \mid \text{the } i\text{-th bit of } |I_C| \text{ is } 1\}$ is in $\mathbf{L}^{\mathbf{NL}} = \mathbf{NL}$ (cf. Proposition 2.4). Consequently, $|I_C| = c_{S_C}(C, 0)c_{S_C}(C, 1) \ldots c_{S_C}(C, m)$ where $m = \lceil \log n \rceil$.
Lower bound. For a function $f \in \mathbf{FL}_{\|}^{\mathbf{NL}}[O(\log n)]$ there exists logspace computable functions g_1 and g_2' such that $g_2'(x) = (G_0, \ldots, G_m)$ and $f(x) =$

$g_1(c_{\mathrm{GAP}}(G_m)\dots c_{\mathrm{GAP}}(G_1)c_{\mathrm{GAP}}(G_0),x)$, where $m =_{\mathrm{def}} \lceil c\log n\rceil$ for suitable $c>0$. (Note that GAP is complete for **NL**, hence we can replace any **NL**-oracle by GAP.) Now let $f'(G_1,\dots,G_k) =_{\mathrm{def}} \sum_{i=1}^{k} c_{\mathrm{GAP}}(G_i)$. In the following we will show that $f \leq_{\text{1-T}}^{log} f'$ and $f' \leq_{\text{1-T}}^{log} \#^{\mathrm{C}}(\{\mathrm{et}\})$, hence $\#^{\mathrm{C}}(\{\mathrm{et}\})$ is $\leq_{\text{1-T}}^{log}$-hard for $\mathbf{FL}_{\|}^{\mathbf{NL}}[O(\log n)]$.
$f \leq_{\text{1-T}}^{log} f'$: Since $f(x) = g_1(c_{\mathrm{GAP}}(G_m)\dots c_{\mathrm{GAP}}(G_0),x) = g_1(\sum_{i=0}^{m} c_{\mathrm{GAP}}(G_i)\cdot 2^i,$ $x) = g_1(\sum_{i=0}^{m}\sum_{j=1}^{2^i} c_{\mathrm{GAP}}(G_i),x)$, there is a logspace computable function g_2, which simulates g_2' in a suitable way, such that

$$g_2(x) = (G_0,G_1,G_1,G_2,\dots,\underbrace{G_m,G_m,\dots,G_m}_{2^m-times})$$

holds. Therefore $f(x) = g_1(f'(g_2(x)),x)$, showing that $f \leq_{\text{1-T}}^{log} f'$.
$f' \leq_{\text{1-T}}^{log} \#^{\mathrm{C}}(\{\mathrm{et}\})$: From every argument G_i construct an $\{\mathrm{et}\}$-circuit C_i' with input variables u_i and z as follows. Every outdegree 2 vertex of G_i becomes an et-gate, the start vertex of G_i becomes the output gate of C_i', the target vertex of G_i becomes an u_i-gate, and every other outdegree 0 vertex of G_i becomes a z-gate. Finally let $C_i =_{\mathrm{def}} C_i' \wedge z$. Obviously, if $G_i \in \mathrm{GAP}$ then $f_{C_i}(u_i,z) = u_i \wedge z$, otherwise $f_{C_i}(u_i,z) = z$. Now construct an $\{\mathrm{et}\}$-circuit C in such a way that $C(u_1,\dots,u_k,z) = \bigwedge_{i=1}^{k} C_i(u_i,z)$. Consequently, f_C has $\sum_{i=1}^{k} c_{\mathrm{GAP}}(G_i)+1$ non-fictive variables and hence $\#^{\mathrm{C}}(\{\mathrm{et}\})(C) = 2^{k-\sum_{i=1}^{k} c_{\mathrm{GAP}}(G_i)}$, i.e., $\sum_{i=1}^{k} c_{\mathrm{GAP}}(G_i) = k - \log_2 \#^{\mathrm{C}}(\{\mathrm{et}\})(C)$ and therefore $f' \leq_{\text{1-T}}^{log} \#^{\mathrm{C}}(\{\mathrm{et}\})$.

4. Let $B \not\subseteq \mathrm{N}$ and $B \subseteq \mathrm{V}$. By inspecting Figure 2.2 we obtain that $\mathrm{V}_2 \subseteq [B] \subseteq \mathrm{V}$. Moreover we know by Proposition 2.12 that $\mathrm{E}_2 = \mathrm{dual}(\mathrm{V}_2) \subseteq \mathrm{dual}([B]) = [\mathrm{dual}(B)] \subseteq \mathrm{dual}(\mathrm{V}) = \mathrm{E}$. Hence $\#^{\mathrm{C}}(\mathrm{dual}(B))$ is $\leq_{\text{1-T}}^{log}$-hard for $\mathbf{FL}_{\|}^{\mathbf{NL}}[O(\log n)]$. By Proposition 3.17 we obtain that $\#^{\mathrm{C}}(B)$ is $\leq_{\text{1-T}}^{log}$-hard for $\mathbf{FL}_{\|}^{\mathbf{NL}}[O(\log n)]$. Since $\mathbf{FL}_{\|}^{\mathbf{NL}}[O(\log n)]$ is closed under $\leq_{\text{1-T}}^{log}$-reductions (use recomputation to show this (cf. [BDG95], Lemma 3.35)) the upper bound follows.

5. Let $B \not\subseteq \mathrm{N}$, $B \not\subseteq \mathrm{D}$, and $B \subseteq \mathrm{L}$. A look at Figure 2.2 shows that $\mathrm{L}_0 \subseteq [B] \subseteq \mathrm{L}$ or $\mathrm{L}_1 \subseteq [B] \subseteq \mathrm{L}$.
Let $\mathrm{L}_0 \subseteq [B] \subseteq \mathrm{L}$. By Theorem 2.11 we obtain $[\{\mathrm{aut}\}] \subseteq [B] \subseteq [\{\mathrm{aut},1\}]$, and by Proposition 3.15 we obtain $\#^{\mathrm{C}}(\{\mathrm{aut}\}) \leq_{\mathrm{m}}^{\log} \#^{\mathrm{C}}(B) \leq_{\mathrm{m}}^{\log} \#^{\mathrm{C}}(\{\mathrm{aut},1\})$.
Upper bound. Let C be an $\{\mathrm{aut},1\}$-circuit with n input variables. There are three different cases. If f_C has a non-fictive variable then $\#_1(C) = 2^{n-1}$. Otherwise f_C is a constant. If $f_C = 0$ ($f_C = 1$, resp.) then $\#_1(C) = 0$ ($\#_1(C) = 2^n$, resp.) Which of these cases takes place can be found out by two queries. To check whether f_C has non-fictive variables use the query "do there exists an i such that $f_C(0^n) \neq f_C(0^{i-1}10^{n-i})$?" and to distinguish the cases $f_C = 0$ and $f_C = 1$ we have additionally to ask "$f_C(0^n) = 1$?". However, by Theorem 3.4 these are $\oplus\mathbf{L}$ queries (notice that $\mathbf{L}^{\oplus\mathbf{L}} = \oplus\mathbf{L}$ by Proposition 2.4).
Lower bound. It is sufficient to prove that there exist logspace computable functions g_1 and g_2 such that $c_{\mathrm{GOAP}}(G) = g_1(\#^{\mathrm{C}}(\{\mathrm{aut}\})(g_2(G)))$, where c_{GOAP} is the characteristic function of GOAP (cf. Theorem 2.5). Given a directed acyclic graph

G whose vertices have outdegree 0 or 2, a start vertex s, and a target vertex t with outdegree 0. We construct an $\{\mathrm{aut}\}$-circuit C with two input variables x and z as follows: Every outdegree 2 vertex becomes an aut-gate, the vertex t becomes an aut-gate with incoming edges from an x-gate and a z-gate, and every outdegree 0 vertex becomes an aut-gate with both incoming edges from a z-gate. The start-vertex s becomes the output gate of C. If there is an odd number of paths from s to t in G then $f_C(x,z) = x \oplus z$ and hence $\#_1(C) = 2$. Otherwise $f_C(x,z) = 0$ and hence $\#_1(C) = 0$. Consequently $c_{\mathrm{GOAP}}(G) = \frac{1}{2} \cdot \#^{\mathrm{C}}(\{\mathrm{aut}\})(C)$, which shows the needed reduction.

Now let $\mathrm{L}_1 \subseteq [B] \subseteq \mathrm{L}$. By Proposition 2.12 we obtain $\mathrm{L}_0 = \mathrm{dual}(\mathrm{L}_1) \subseteq \mathrm{dual}([B]) = [\mathrm{dual}(B)] \subseteq \mathrm{dual}(\mathrm{L}) = \mathrm{L}$, and by the above we get that $\#^{\mathrm{C}}(\mathrm{dual}(B))$ is $\leq^{log}_{1\text{-T}}$-hard for $\mathbf{FL}_{||}^{\oplus\mathbf{L}}[1]$. With Proposition 3.17 we conclude that $\#^{\mathrm{C}}(B)$ is $\leq^{log}_{1\text{-T}}$-hard for $\mathbf{FL}_{||}^{\oplus\mathbf{L}}[1]$. Since $\mathbf{FL}_{||}^{\oplus\mathbf{L}}[2]$ is closed under $\leq^{log}_{1\text{-T}}$-reductions, we conclude by Proposition 3.15 that $\#^{\mathrm{C}}(B) \in \mathbf{FL}_{||}^{\oplus\mathbf{L}}[2]$.

6. Let $B \not\subseteq \mathrm{D}$, $B \not\subseteq \mathrm{V}$, $B \not\subseteq \mathrm{E}$, $B \not\subseteq \mathrm{L}$, and $B \subseteq \mathrm{R}_1$. An inspection of Figure 2.2 shows that $[B] \supseteq \mathrm{S}_{10}$ or $[B] \supseteq \mathrm{S}_{00}$. By Lemma 3.18 we obtain that $\#^{\mathrm{C}}(B)$ is $\leq^{log}_{1\text{-T}}$-complete for $\#\mathbf{P}$. Because of $B \subseteq \mathrm{R}_1$ we have $\#^{\mathrm{C}}(B)(C) > 0$ for every B-circuit C. Hence $\#^{\mathrm{C}}(B)$ cannot be $\leq^{\mathrm{log}}_{\mathrm{m}}$-complete for $\#\mathbf{P}$.

7. Let $B \not\subseteq \mathrm{D}$, $B \not\subseteq \mathrm{V}$, $B \not\subseteq \mathrm{E}$, $B \not\subseteq \mathrm{L}$, and $B \subseteq \mathrm{M}$. As above we obtain that $\#^{\mathrm{C}}(B)$ is $\leq^{log}_{1\text{-T}}$-complete for $\#\mathbf{P}$. Assume that $\#^{\mathrm{C}}(B)$ is $\leq^{\mathrm{log}}_{\mathrm{m}}$-complete for $\#\mathbf{P}$. For an arbitrary $A \in \mathbf{NP}$ there exists an $f \in \#\mathbf{P}$ such that $x \in A$ iff $f(x) > 0$ for every x. Since $\#^{\mathrm{C}}(B)$ is $\leq^{\mathrm{log}}_{\mathrm{m}}$-complete for $\#\mathbf{P}$ there exists a logspace computable function h such that $f(x) = \#^{\mathrm{C}}(B)(h(x))$ for every x. Consequently, $x \in A$ iff $\#^{\mathrm{C}}(B)(h(x)) > 0$ iff $h(x) \in \mathrm{SAT}^{\mathrm{C}}(B)$. Because of $B \subseteq \mathrm{M}$ we obtain $A \in \mathbf{P}$ by Theorem 3.8.

8. Let $B \not\subseteq \mathrm{D}$, $B \not\subseteq \mathrm{L}$, $B \not\subseteq \mathrm{R}_1$, and $B \not\subseteq \mathrm{M}$. An inspection of Figure 2.2 shows that $[B] \supseteq \mathrm{S}_1$. By Lemma 3.18 we obtain that $\#^{\mathrm{C}}(B)$ is $\leq^{\mathrm{log}}_{\mathrm{m}}$-complete for $\#\mathbf{P}$. □

In the case that $\mathrm{L}_0 \subseteq [B] \subseteq \mathrm{L}$ or $\mathrm{L}_1 \subseteq [B] \subseteq \mathrm{L}$ there is an oddity left. Here the lower bound and the upper bound do not coincide in contrast to all other cases discussed so far. Therefore one might ask whether it is possible to improve the situation. For this assume that the upper bound is not optimal and we have to improve it to $\mathbf{FL}_{||}^{\oplus\mathbf{L}}[1]$. Now note that for any $\{aut, 1\}$-circuit $C(x_1, \ldots, x_n)$ one of the three cases $\#_1(C) = 2^{n-1}$, $\#_1(C) = 0$ or $\#_1(C) = 2^n$ holds. Therefore we had to handle 3 cases with one answer by a $\oplus\mathbf{L}$-oracle, since as long as $\mathbf{L} \neq \oplus\mathbf{L}$ holds, it is not clear how we can gain additional information (like satisfiability, circuit value, etc) in the logspace base computation. Hence it seems to be a complicated task to improve the upper bound.

Another possibility would be to improve the lower bound. For this we have to show that $\#^{\mathrm{C}}(\{\mathrm{aut}, 1\})$ is $\leq^{log}_{1\text{-T}}$-hard for $\mathbf{FL}_{||}^{\oplus\mathbf{L}}[2]$. For this note that two oracle queries result in four possible computation paths. Therefore we have to code four possible different situations into one $\{\mathrm{aut}, 1\}$-circuit and ask for the number of satisfying assignments of this circuit. As mentioned above we only have three different possibilities for the number of satisfying assignments of an $\{\text{aut}, 1\}$-circuit. Therefore it is not clear how to do the reduction.

In both cases it is not clear how to improve the upper bound respectively lower bound and hence we are forced to give a different lower bound and upper bound in this special case.

3.6 The threshold problem

In this section we will study the complexity of the *threshold problem*, which is defined for every finite set B of Boolean functions as follows:

PROBLEM: $\text{THR}^{\text{C}}(B)$
INSTANCE: A B-circuit C and $k \geq 0$
QUESTION: Is $\#_1(C) > k$?

Obviously, $\text{THR}^{\text{C}}(B) \in \mathbf{PP}$.

The following result on the complexity of $\text{THR}^{\text{C}}(B)$ can easily be derived from the literature, where it is proved for the special case of $\{\text{et}, \text{vel}, \text{non}\}$-formulas.

Theorem 3.20. *[Gil77] For a finite set B of Boolean functions, if $[B] = \text{BF}$ then $\text{THR}^{\text{C}}(B)$ is even $\leq_{\text{m}}^{\log}$-complete for $\mathbf{PP}$.*

Note that this completeness result also holds for $\{\text{et}, \text{vel}, \text{non}\}$-formulas in 3-CNF.

Now we are ready to prove the main theorem on the complexity of $\text{THR}^{\text{C}}(B)$:

Theorem 3.21. *Let be B a finite set of Boolean functions.*

if *$((B \subseteq \text{N})$* ***or*** *$(B \subseteq \text{D}))$* ***then***
$\text{THR}^{\text{C}}(B) \in \mathbf{L}$
else if *$((B \subseteq \text{E})$* ***or*** *$(B \subseteq \text{V}))$* ***then***
$\text{THR}^{\text{C}}(B)$ *is $\leq_{\text{m}}^{\log}$-complete for* $\mathbf{NL}$
else if *$(B \subseteq \text{L})$* ***then***
$\text{THR}^{\text{C}}(B)$ *is $\leq_{\text{m}}^{\log}$-complete for* $\oplus\mathbf{L}$
else
$\text{THR}^{\text{C}}(B)$ *is $\leq_{\text{m}}^{\log}$-complete for* $\mathbf{PP}$.

There exists an algorithm which decides which of the cases above takes place.

Proof. 1. If $B \subseteq \text{N}$ or $B \subseteq \text{D}$ then, by Theorem 3.19, we have $\#^{\text{C}}(B) \in \mathbf{FL}$ and consequently $\mathbf{PP}(B) \in \mathbf{L}$.

2. Let $B \not\subseteq \text{N}$ and $B \subseteq \text{E}$. By Theorem 3.19 we obtain $\#^{\text{C}}(B) \in \mathbf{FL}_{\|}^{\mathbf{NL}}[O(\log n)]$ and hence $\text{THR}^{\text{C}}(B) \in \mathbf{L}^{\mathbf{NL}} = \mathbf{NL}$ (cf. Proposition 2.4).
For the lower bound it is sufficient to prove $\overline{GAP} \leq_{\text{m}}^{\log} \text{THR}^{\text{C}}(\{\text{et}\})$ (cf. Theorem 2.5). As in Item 3 (second part) of the proof of Theorem 3.19 we construct for a graph G an $\{\text{et}\}$-circuit C with the following property: If $G \in \text{GAP}$ then $f_C(u, z) = u \wedge z$, and $f_C(u, z) = z$ otherwise. Hence, if $G \in \overline{\text{GAP}}$ then $\#_1(C) = 2$ and $(C, 2) \in \text{THR}^{\text{C}}(\{\text{et}\})$. Otherwise $\#_1(C) = 1$ and $(C, 2) \notin \text{THR}^{\text{C}}(\{\text{et}\})$.

3. The case $B \not\subseteq \mathrm{N}$ and $B \subseteq \mathrm{V}$ is done in an analogous manner.

4. The case $B \not\subseteq \mathrm{N}$, $B \not\subseteq \mathrm{D}$, and $B \subseteq \mathrm{L}$ is also done analogously using the construction in Item 5 of the proof of Theorem 3.19 and the problem GOAP.

5. Let $B \not\subseteq \mathrm{D}$, $B \not\subseteq \mathrm{V}$, $B \not\subseteq \mathrm{E}$, and $B \not\subseteq \mathrm{L}$. An inspection of Figure 2.2 shows that $[B] \supseteq \mathrm{S}_{10}$ or $[B] \supseteq \mathrm{S}_{00}$ For a set $A \in \mathbf{PP}$ there exists a $f \in \#\mathbf{P}$ such that $(x, k) \in A$ iff $f(x) \geq k$. By Lemma 3.18 there exist logspace computable functions g and h such that $f(x) = \#_1(h(x)) - g(x)$. Consequently, $(x, k) \in A$ iff $\#_1(h(x)) \geq g(x) + k$ iff $(h(x), g(x) + k) \in \mathrm{THR}^{\mathrm{C}}(B)$. □

3.7 Tree-like circuits

In the previous sections we studied problems related to circuits. What can be said about tree-like B-circuits, i.e., B-formulas?

Informally we can say that all completeness results for complexity classes beyond $\mathbf{P}$ remain valid for B-formulas. For classes inside $\mathbf{P}$ this is not true since the evaluation problem of B-formulas can be easier than the evaluation problem for B-circuits. It is known that the formula value problem of $\{\mathrm{et}, \mathrm{vel}, \mathrm{non}\}$-formulas is complete for $\mathbf{NC^1}$, the class of problems which can be solved by $\{\mathrm{et}, \mathrm{vel}, \mathrm{non}\}$-circuits of polynomial size (i.e., number of gates), bounded fan-in and logarithmic depth (i.e., length of the longest path from an input-gate to the output-gate (cf. [Bus87])).

As a brief example of how to deal with formulas we will shortly study the satisfiability problem and the tautology problem for B-formulas, defined as follows:

PROBLEM: $\mathrm{SAT}^{\mathrm{F}}(B)$
INSTANCE: A B-formula $H(x_1, \ldots, x_n)$
QUESTION: Exists a $a \in \{0,1\}^n$ such that $f_H(a) = 1$?

PROBLEM: $\mathrm{TAUT}^{\mathrm{F}}(B)$
INSTANCE: A B-formula $H(x_1, \ldots, x_n)$
QUESTION: For all $a \in \{0,1\}^n$ hold $f_H(a) = 1$?

One might think that it is easy to prove that if $\mathrm{S}_1 \subseteq [B]$, then $\mathrm{SAT}^{\mathrm{F}}(B)$ is $\leq_{\mathrm{m}}^{\log}$-complete for $\mathbf{NP}$ by using the techniques of Proposition 3.7 and Theorem 3.8. But there is one major drawback. Let $B = \{g\}$, where $g(x, y) = x \wedge \overline{y}$ is a base of S_1. Clearly $g(x, g(x, y)) = et(x, y)$. Therefore we have a B-formula representing the et-function, but this B-formula contains the variable x more than one time. If we simply use this B-formula to transform an $\{\mathrm{et}, \mathrm{vel}, \mathrm{non}\}$-formula into a B-formula an exponential "blow-up" of the resulting formula can happen. (Note that we were able to avoid this problem in the $\mathrm{SAT}^{\mathrm{C}}(B)$-case by using the fact that all gates in a B-circuit can have a fan-out of more than 1.) Hence we cannot hope to find a suitable reduction function, which produces the needed B-formula. Because of this, it is not clear how to show that $\mathrm{SAT} \leq_{\mathrm{m}}^{\log} \mathrm{SAT}^{\mathrm{F}}(\mathrm{S}_1)$ holds.

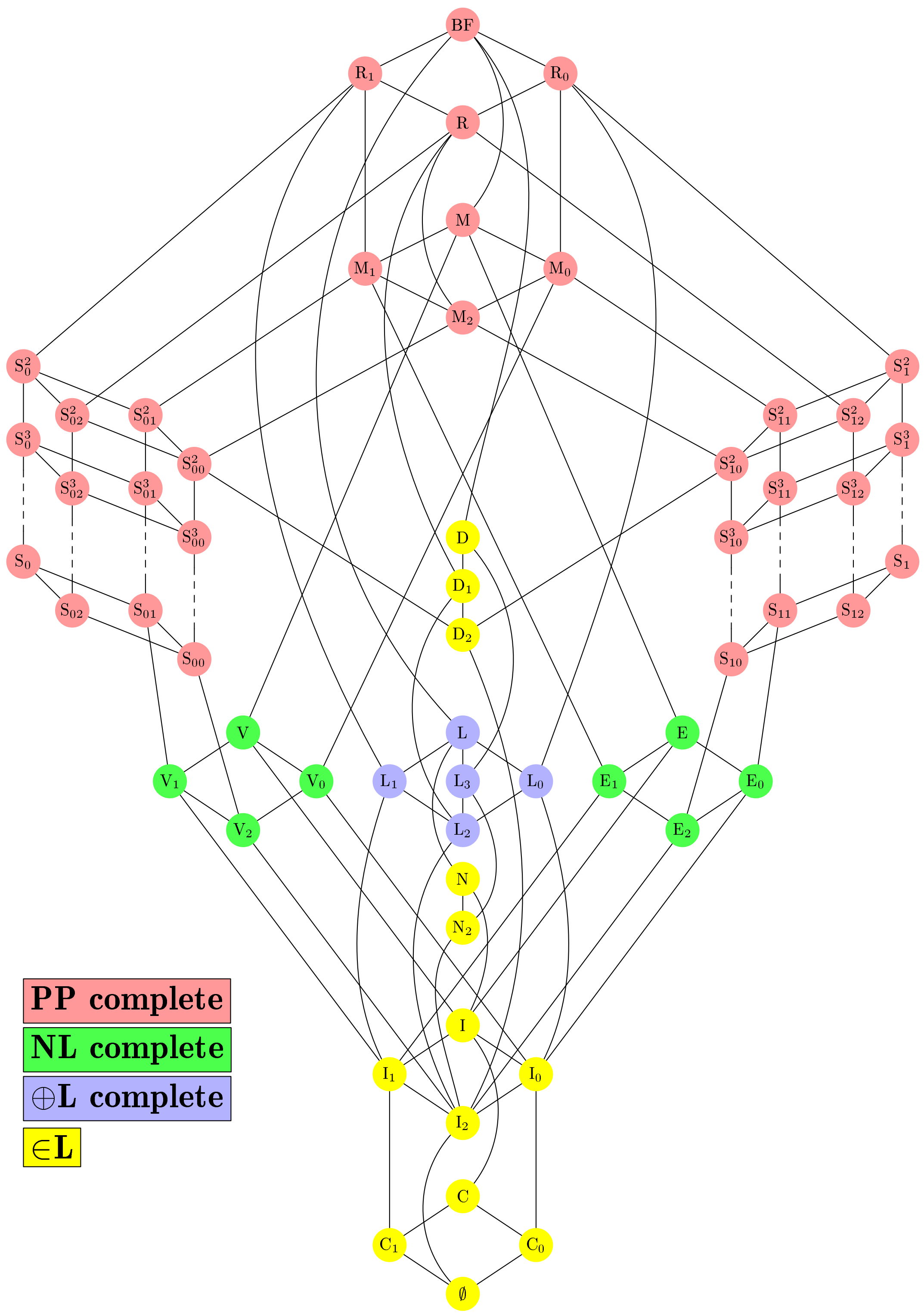


Fig. 3.7. The complexity of $\mathrm{THR}^{\mathrm{C}}(B)$.

For this problem there are two completely different ways out. First we can use a restricted version of SAT, being still complete for **NP**, as a starting point, such that this problem does not occur. We will use this technique in Section 5.3, where we focus on B-formulas. For this we use the problem 3-SAT to avoid the blow-up by inserting brackets into CNF-formulas in a special way.

The other loophole hinges on a beautiful result achieved by Lewis (see [Lew79]):

Proposition 3.22 ([Lew79]). *Let B be a complete set of Boolean functions, i.e., $[B] = \mathrm{BF}$. Then the following three statements hold:*

1. *There is a B-formula $N(x) \equiv \overline{x}$ having only one occurrence of x.*
2. *There is a B-formula $E(x, y) \equiv x \wedge y$ having only one occurrence of each of the variables x, y.*
3. *There is a B-formula $V(x, y) \equiv x \vee y$ having only one occurrence of each of the variables x, y.*

Clearly by using such formulas we can avoid the problem of exponential blow-up and the resulting B-formula is computable in logspace. Now, by using Proposition 3.22 we can proof the following lemma, which was stated the first time in [Lew79]:

Lemma 3.23 ([Lew79]). *Let B be a finite set of Boolean functions. If $\mathrm{S}_1 \subseteq [B]$ then $\mathrm{SAT}^{\mathrm{F}}(B)$ is $\leq_{\mathrm{m}}^{\log}$-complete for* **NP**.

Proof. Use the B-formulas N, E and V from Proposition 3.22 to show $\mathrm{SAT} \leq_{\mathrm{m}}^{\log} \mathrm{SAT}^{\mathrm{F}}(B)$. For this note that $[B \cup \{1\}] = \mathrm{BF}$. Using Proposition 3.7, adapted to the formula case, the statement follows. □

Using this result we can give the following dichotomy result for $\mathrm{SAT}^{\mathrm{F}}(B)$, which shows the succinctness of circuits.

Theorem 3.24. *Let B be a finite set of Boolean functions.*

if *($\mathrm{S}_1 \subseteq [B]$)* ***then***
$\mathrm{SAT}^{\mathrm{F}}(B)$ *is $\leq_{\mathrm{m}}^{\log}$-complete for* **NP**
else
$\mathrm{SAT}^{\mathrm{F}}(B) \in \mathbf{L}$.

Proof. The first part of this theorem is given by Lemma 3.23. Now following Theorem 3.8 we have only to prove that $\mathrm{SAT}^{\mathrm{F}}(B) \in \mathbf{L}$ in the cases that $\mathrm{SAT}^{\mathrm{C}}(B)$ is **NL**-complete, $\oplus$**L**-complete or **P**-complete, because $\mathrm{SAT}^{\mathrm{F}}(B) \leq_{\mathrm{m}}^{\log} \mathrm{SAT}^{\mathrm{C}}(B)$ and hence if $\mathrm{SAT}^{\mathrm{C}}(B) \in \mathbf{L}$ also $\mathrm{SAT}^{\mathrm{F}}(B) \in \mathbf{L}$. Using Lemma 4.11 from Chapter 4 this can be shown as follows:

Case 1: $[B] = \mathrm{V}_0$, $[B] = \mathrm{V}$, $[B] = \mathrm{E}_0$, $[B] = \mathrm{E}$ or $\mathrm{S}_{11} \subseteq [B] \subseteq \mathrm{M}$
Let H be a B-formula. We know that $H(x_1, \ldots, x_n) \in \mathrm{SAT}^{\mathrm{F}}(B)$ iff $(H, 1^n) \in \mathrm{VAL}^{\mathrm{F}}(B)$ (cf. Proposition 3.6). Since $\{\mathrm{et}, \mathrm{vel}, \mathrm{non}\}$ is a complete set of Boolean functions, there is by Proposition 3.22 a logspace computable $\{\mathrm{et}, \mathrm{vel}, \mathrm{non}\}$-formula $H'(x_1, \ldots, x_n)$ such that $f_H = f_{H'}$. Hence $H \in \mathrm{SAT}^{\mathrm{F}}(B)$ iff

$(H', 1^n) \in \mathrm{VAL}^{\mathrm{F}}(\{\mathrm{et}, \mathrm{vel}, \mathrm{non}\})$, which shows that $\mathrm{SAT}^{\mathrm{F}}(B) \leq_{\mathrm{m}}^{\log} \mathrm{VAL}^{\mathrm{F}}(\{\mathrm{et}, \mathrm{vel}, \mathrm{non}\})$. Using the result of Buss ([Bus87]) that $\mathrm{VAL}^{\mathrm{F}}(\{\mathrm{et}, \mathrm{vel}, \mathrm{non}\}) \in \mathbf{NC^1}$ and the fact that the logspace-closure of $\mathbf{NC^1}$ is clearly $\mathbf{L}$ we conclude that $\mathrm{SAT}^{\mathrm{F}}(B) \in \mathbf{L}$.

Case 2: $[B] = \mathrm{L}_0$ or $[B] = \mathrm{L}$

Let $H(x_1, \ldots, x_n)$ be an arbitrary B-formula. Since B is linear, the Boolean function represented by H can be written as $f_H(x_1, \ldots, x_n) = a_0 \oplus (a_1 \wedge x_1) \oplus \cdots \oplus (a_n \wedge x_n)$, where $a_0, \ldots, a_n \in \{0, 1\}$. Note that $a_0 = f_H(0^n)$ and $a_i = f_H(0^{n-1}10^{i-n}) \oplus a_0$ for $1 \leq i \leq n$. Since $H \in \mathrm{SAT}^{\mathrm{F}}(B)$ iff $a_0 = 1$ or there is an i such that $a_i = 1$, we conclude that $\mathrm{SAT}^{\mathrm{F}}(B) \in \mathbf{L^L} = \mathbf{L}$, because the formula value problem can be solved by a $\mathbf{L}$-computation in this case (cf. Lemma 4.11). □

Note that we also can make use of Lemma 4.11 in the cases $[B] = \mathrm{V}_0$, $[B] = \mathrm{V}$, $[B] = \mathrm{E}_0$ or $[B] = \mathrm{E}$.

By adapting Proposition 3.9 and the fact that $\mathbf{L}$ is closed under complement we achieve the following dichotomy theorem for $\mathrm{TAUT}^{\mathrm{F}}(B)$.

Theorem 3.25. *Let B be a finite set of Boolean functions.*

if *($\mathrm{S}_0 \subseteq [B]$)* ***then***
$\mathrm{TAUT}^{\mathrm{F}}(B)$ *is* $\leq_{\mathrm{m}}^{\log}$*-complete for* **coNP**
else
$\mathrm{TAUT}^{\mathrm{F}}(B) \in \mathbf{L}$.

Similarly we can achieve results for quantified formulas, counting functions related to B-formulas and the threshold problem.

4. Equivalence- and isomorphism-problems

4.1 Introduction

Besides the problems in Chapter 3, other interesting problems in the theory of Boolean functions were studied. Two of them are the properties of being equal or isomorphic (i.e. is there a permutation of the variables of the first function, such that this modified function is equal to the second function). This question has a history going back to the 19$^{\text{th}}$ century (see [BRS98] for a list of early references). Moreover in [BRS98] other generalized equality relations for propositional formulas were studied.

In the case of unrestricted formulas it is known that the Boolean equivalence-problem for formulas and circuits is **coNP**-complete, whereas only very weak lower and upper bounds for the Boolean isomorphism-problem are known. By a reduction from the tautology problem, which is **coNP**-complete, a lower bound for the isomorphism problem can be easily derived. An upper bound for the isomorphism problem is clearly $\mathbf{\Sigma_2^p}$, since a $\mathbf{\Sigma_2^p}$-algorithm can existentially guess a permutation of variables and universally check the functions for equality.
In [AT00] it was shown that the complement of the Boolean isomorphism-problem for formulas and circuits has an one-round interactive proof, where the verifier has access to an **NP** oracle. In that paper a better lower bound for the isomorphism-problem of unrestricted propositional formulas was given, too. More precisely Agrawal and Thierauf have shown that UOCLIQUE $\leq_{\mathrm{m}}^{\mathrm{p}}$ ISO$^{\mathrm{F}}$ holds, where by ISO$^{\mathrm{F}}$ we denote the isomorphism-problem of unrestricted propositional formulas and by UOCLIQUE the problem of checking whether the largest clique in a graph is unique. It is known that UOCLIQUE is $\leq_{\mathrm{m}}^{\mathrm{p}}$-hard for 1-**NP**, a superclass of **coNP**, where 1-**NP** denotes the class of all problems, whose solution can be found on exactly one path in nondeterministic polynomial time. Additionally it is known that the graph isomorphism-problem can be reduced to the Boolean isomorphism-problem for unrestricted Boolean formulas (see [BRS98]).

In this chapter we focus on the complexity of checking whether two given B-formulas represent the same Boolean function (equivalence-problem for B-formulas) or represent isomorphic Boolean functions (isomorphism-problem for B-formulas). Moreover these problems are studied for B-circuits too. We restrict ourselves to these two equality relations of Boolean functions, because they correlate with a restricted superposition (note that superposition includes the permutation of variables) and therefore they nicely fit into the context of Post's closed classes.

We give, where possible, tight upper and lower bounds for the isomorphism-problem of B-formulas and B-circuits. In all other cases we show the **coNP**-hardness for the isomorphism-problem, which is as good as the trivial lower bound in the unrestricted case. Note that the upper bound $\mathbf{\Sigma_2^p}$ holds for our B-formulas and B-circuits as well, since we restrict ourselves to possibly non-complete sets of Boolean functions used as connectors for our generalized propositional formulas or circuits. For the equivalence-problem we always give tight upper and lower bounds, showing that this problem is in **L**, **NL**-complete, $\oplus$**L**-complete or **coNP**-complete, depending on the used set B.

4.2 Preliminaries

Definition 4.1. Let B be a finite set of Boolean functions, $C(x_1, \dots, x_n) = (P, E, o, \alpha, \beta, \gamma)$ be a B-circuit and $\pi\colon \{1, \dots, n\} \to \{1, \dots, n\}$ be a permutation. For convenience we use $\pi(x_i) =_{\text{def}} x_{\pi(i)}$ for a propositional variable x_i, where $1 \le i \le n$. We define $\pi(C) =_{\text{def}} (P, E, o, \alpha, \beta', \gamma)$, where $\beta'(v) =_{\text{def}} \beta(v)$ iff $\beta(v) \notin \{x_1, \dots, x_n\}$ and $\beta'(v) =_{\text{def}} \pi(\beta(v))$ otherwise. As a shortcut for this definition we use $\pi(C(x_1, \dots, x_n)) = C(x_{\pi(1)}, \dots, x_{\pi(n)})$, which symbolizes that in $\pi(C)$ the variables x_i are replaced by $x_{\pi(i)}$.

The *Boolean equivalence- and isomorphism relation* for B-circuits is introduced as follows:

$$
\begin{aligned}
C_1 \equiv C_2 \Leftrightarrow_{\text{def}} \quad & f_{C_1}(a_1, \dots, a_n) = f_{C_2}(a_1, \dots, a_n) \text{ for all } a_1, \dots, a_n \in \{0,1\}.\\
C_1 \cong C_2 \Leftrightarrow_{\text{def}} \quad & \text{There exists a permutation } \pi\colon \{1, \dots, n\} \to \{1, \dots, n\}\\
& \text{such that } \pi(C_1) \equiv C_2.
\end{aligned}
$$

These definitions will be analogously used for B-formulas.

Note that in both cases the sets of input-variables of C_1 and C_2 are equal. This can be easily achieved by adding syntactically fictive variables when needed. By this definition it is clear that $H_1 \equiv H_2$ iff for all assignments I it holds that $I \models H_1 \iff I \models H_2$. Moreover note that $C_1 \equiv C_2$ iff $\text{dual}(C_1) \equiv \text{dual}(C_2)$ and $H_1 \equiv H_2$ iff $\text{dual}(H_1) \equiv \text{dual}(H_2)$.

Now we are ready to define the *Boolean equality-problem* for B-circuits:

PROBLEM: $\text{EQ}^{\text{C}}(B)$
INSTANCE: Two B-circuits C_1 and C_2
QUESTION: Is $C_1 \equiv C_2$?

Similarly we define the *Boolean equality-problem* $\text{EQ}^{\text{F}}(B)$ for B-formulas. Next we introduce the isomorphism-problem for B-circuits:

PROBLEM: $\text{ISO}^{\text{C}}(B)$

INSTANCE: Two B-circuits C_1 and C_2
QUESTION: Is $C_1 \cong C_2$?

Analogously we define the *Boolean isomorphism-problem* $\mathrm{ISO}^{\mathrm{F}}(B)$ for B-formulas.

Since we must be able to talk about assignments of $\pi(H)$ for a B-formula H and a permutation π, we define the assignment $\pi(I)$ as follows:

Definition 4.2. Let $\pi\colon \{1,\ldots,n\} \to \{1,\ldots,n\}$ be a permutation and $I\colon \{x_1, \ldots, x_n\} \to \{0,1\}$ be an assignment such that $I(x_i) = a_i$, where $a_i \in \{0,1\}$ and $1 \leq i \leq n$. We define the assignment $\pi(I)$ by $\pi(I)(x_{\pi(i)}) =_{\mathrm{def}} a_i$.

Proposition 4.3. *Let $H_1(x_1,\ldots,x_n) \cong H_2(x_1,\ldots,x_n)$ via $\pi\colon \{1,\ldots,n\} \to \{1,\ldots,n\}$ and I be an assignment which is compatible with H_1 and H_2. Then the following two statements hold:*

1. $I \models \pi(H_1)$ *iff* $I \models H_2$
2. $I \models H_1$ *iff* $\pi(I) \models H_2$

Proof. Let $H_1 \cong H_2$ via permutation π. Hence we know that $\pi(H_1) \equiv H_2$ and therefore $f_{\pi(H_1)}(a_1,\ldots,a_n) = f_{H_2}(a_1,\ldots,a_n)$ for all $a_1,\ldots,a_n \in \{0,1\}$. Because of this we conclude for an assignment I that $I \models \pi(H_1)$ iff $I \models H_2$.

Now let I, I' be assignments which are defined as follows: $I(x_i) =_{\mathrm{def}} a_i$ and $I'(\pi(x_i)) =_{\mathrm{def}} a_i$, where $1 \leq i \leq n$. Clearly $I \models H_1$ iff $f_{H_1}(a_1,\ldots,a_n) = 1$ iff $I' \models \pi(H_1)$, because we get $\pi(H_1)$ from H_1 by substituting the variables x_i by $\pi(x_i)$. Because of the first statement we conclude that $I' \models \pi(H_1)$ iff $I' \models H_2$ and therefore $I \models H_1$ iff $I' \models H_2$. Finally note that $I' = \pi(I)$, which proves the second statement. □

Clearly this proposition holds for B-formulas as well as for B-circuits.

Example 4.4. Let $H_1(x_1,x_2) = x_1 \wedge (\neg x_1 \vee \neg x_2)$ and $H_2(x_1,x_2) = \neg x_1 \wedge x_2$. Obviously $H_1 \not\equiv H_2$, since $(1,0) \models H_1$ and $(1,0) \not\models H_2$. Note that $(1,0)$ ($(0,1)$, resp.) is the only satisfying assignment for H_1 (H_2, resp.). The permutation $\pi(1) = 2$ and $\pi(2) = 1$ shows that $H_1 \cong H_2$. Now we know that $\pi(H_1) = x_2 \wedge (\neg x_2 \vee \neg x_1)$ and $(0,1)$ is the only model for $\pi(H_1)$ by the first part of Proposition 4.3. Since $(1,0)$ is the only model of H_1, we know that $\pi((1,0)) = (0,1)$ is the only model of H_2 by the second part of Proposition 4.3.

The next proposition shows, that if we permute the variables of a given B-circuit or B-formula then the number of satisfying assignments remains equal.

Proposition 4.5. *Let $H_1(x_1,\ldots,x_n)$ and $H_2(x_1,\ldots,x_n)$ be B-formulas such that $H_1 \cong H_2$. Then $\#_1(H_1) = \#_1(H_2)$ and $\#_0(H_1) = \#_0(H_2)$ hold.*

Proof. Let $H_1(x_1,\ldots,x_n) \cong H_2(x_1,\ldots,x_n)$. Hence there exists a permutation $\pi\colon \{1,\ldots,n\} \to \{1,\ldots,n\}$ such that $\pi(H_1) \equiv H_2$. By Proposition 4.3 we know that $I \models H_1$ iff $\pi(I) \models H_2$. Now we conclude that $|\{I \mid I \models H_1\}| = |\{\pi(I) \mid \pi(I) \models H_2\}| = |\{I' \mid I' \models H_2\}|$. Therefore $\#_1(H_1) = \#_1(H_2)$. By the facts

$\#_0(H_1) = 2^n - \#_1(H_1)$ and $\#_0(H_2) = 2^n - \#_1(H_2)$ the second statement follows immediately. $\square$

It is obvious that the proof of Proposition 4.5 works for B-circuits too. Note that the opposite direction of Proposition 4.5 does not hold, as $x \oplus y \ncong \neg(x \oplus y)$ and $\#_1(x \oplus y) = \#_1(\neg(x \oplus y)) = 2$ shows.

Next we give an upper bound for the equivalence-problem of B-formulas and B-circuits. Later we will see that this upper bound is tight for B-circuits and B-formulas in some cases. Moreover we show that the equivalence-problem and the isomorphism-problem for B-circuits and $\mathrm{dual}(B)$-circuits (B-formulas and $\mathrm{dual}(B)$-formulas, resp.) is of equal complexity.

Proposition 4.6. *Let B be a finite set of Boolean functions. Then the following three statements hold:*

1. $\mathrm{EQ}^{\mathrm{F}}(B) \leq_{\mathrm{m}}^{\log} \mathrm{EQ}^{\mathrm{C}}(B)$ *and* $\mathrm{ISO}^{\mathrm{F}}(B) \leq_{\mathrm{m}}^{\log} \mathrm{ISO}^{\mathrm{C}}(B)$*,*
2. $\mathrm{EQ}^{\mathrm{F}}(B) \in \mathbf{coNP}$ *and* $\mathrm{EQ}^{\mathrm{C}}(B) \in \mathbf{coNP}$*,*
3. $\mathrm{EQ}^{\mathrm{F}}(B) \equiv_{\mathrm{m}}^{\log} \mathrm{EQ}^{\mathrm{F}}(\mathrm{dual}(B))$ *and* $\mathrm{EQ}^{\mathrm{C}}(B) \equiv_{\mathrm{m}}^{\log} \mathrm{EQ}^{\mathrm{C}}(\mathrm{dual}(B))$*,*
4. $\mathrm{ISO}^{\mathrm{F}}(B) \equiv_{\mathrm{m}}^{\log} \mathrm{ISO}^{\mathrm{F}}(\mathrm{dual}(B))$ *and* $\mathrm{ISO}^{\mathrm{C}}(B) \equiv_{\mathrm{m}}^{\log} \mathrm{ISO}^{\mathrm{C}}(\mathrm{dual}(B))$*.*

Proof. Statement 1 follows directly from Definition 2.13, since B-formulas are defined to be fan-out 1 B-circuits.

For Statement 2 note that $\mathrm{EQ}^{\mathrm{C}}(B)$ is in $\mathbf{coNP}$, since we can guess an assignment nondeterministically and accept iff the values of the two evaluated circuits coincide for the guessed assignment. Because $\mathbf{coNP}$ is closed under logspace reductions $\mathrm{EQ}^{\mathrm{F}}(B) \in \mathbf{coNP}$ follows from Statement 1.

For Statement 3 we show that $\mathrm{EQ}^{\mathrm{F}}(B) \leq_{\mathrm{m}}^{\log} \mathrm{EQ}^{\mathrm{F}}(\mathrm{dual}(B))$ ($\mathrm{EQ}^{\mathrm{C}}(B) \leq_{\mathrm{m}}^{\log} \mathrm{EQ}^{\mathrm{C}}(\mathrm{dual}(B))$, resp.) by $g((H_1, H_2)) =_{\mathrm{def}} (\mathrm{dual}(H_1), \mathrm{dual}(H_2))$ ($g((C_1, C_2)) =_{\mathrm{def}} (\mathrm{dual}(C_1), \mathrm{dual}(C_2))$, resp.). First note that $g \in \mathbf{FL}$, since B is a finite set and $\mathrm{dual}(H_1)$ as well as $\mathrm{dual}(H_2)$ ($\mathrm{dual}(C_1)$ as well as $\mathrm{dual}(C_2)$, resp.) can be calculated by simply replacing the function-symbols of H_1 and H_2 (gate-types of C_1 and C_2, resp.) by the function-symbol (gate-type, resp.) of their dual function. Clearly $\mathrm{dual}(H_1)$ and $\mathrm{dual}(H_2)$ are $\mathrm{dual}(B)$-formulas ($\mathrm{dual}(C_1)$ and $\mathrm{dual}(C_2)$ are $\mathrm{dual}(B)$-circuits, resp.). Let f_{H_1} and f_{H_2} (f_{C_1} and f_{C_2}, resp.) be n-ary Boolean functions, then $f_{H_1}(\alpha) = f_{H_2}(\alpha)$ iff $f_{\mathrm{dual}(H_1)}(\alpha) = \mathrm{dual}(f_{H_1})(\alpha) = \mathrm{dual}(f_{H_2})(\alpha) = f_{\mathrm{dual}(H_2)}(\alpha)$ ($f_{C_1}(\alpha) = f_{C_2}(\alpha)$ iff $f_{\mathrm{dual}(C_1)}(\alpha) = \mathrm{dual}(f_{C_1})(\alpha) = \mathrm{dual}(f_{C_2})(\alpha) = f_{\mathrm{dual}(C_2)}(\alpha)$, resp.) for all $\alpha \in \{0,1\}^n$, showing that $\mathrm{EQ}^{\mathrm{F}}(B) \leq_{\mathrm{m}}^{\log} \mathrm{EQ}^{\mathrm{F}}(\mathrm{dual}(B))$ ($\mathrm{EQ}^{\mathrm{C}}(B) \leq_{\mathrm{m}}^{\log} \mathrm{EQ}^{\mathrm{C}}(\mathrm{dual}(B))$, resp.) holds.
For the other direction note that for an arbitrary B-formula H (B-circuit C, resp.) it holds that $\mathrm{dual}(\mathrm{dual}(H)) = H$ ($\mathrm{dual}(\mathrm{dual}(C)) = C$, resp.). By using the same argument as above, clearly $\mathrm{EQ}^{\mathrm{F}}(\mathrm{dual}(B)) \leq_{\mathrm{m}}^{\log} \mathrm{EQ}^{\mathrm{F}}(\mathrm{dual}(\mathrm{dual}(B))) = \mathrm{EQ}^{\mathrm{F}}(B)$ ($\mathrm{EQ}^{\mathrm{C}}(\mathrm{dual}(B)) \leq_{\mathrm{m}}^{\log} \mathrm{EQ}^{\mathrm{C}}(\mathrm{dual}(\mathrm{dual}(B))) = \mathrm{EQ}^{\mathrm{C}}(B)$, resp.). Therefore $\mathrm{EQ}^{\mathrm{F}}(B) \equiv_{\mathrm{m}}^{\log} \mathrm{EQ}^{\mathrm{F}}(\mathrm{dual}(B))$ and $\mathrm{EQ}^{\mathrm{C}}(B) \equiv_{\mathrm{m}}^{\log} \mathrm{EQ}^{\mathrm{C}}(\mathrm{dual}(B))$.

To prove Statement 4 we use the fact that $\mathrm{ISO}^{\mathrm{F}}(B) \leq_{\mathrm{m}}^{\log} \mathrm{ISO}^{\mathrm{F}}(\mathrm{dual}(B))$ by $g(H_1, H_2) =_{\mathrm{def}} (\mathrm{dual}(H_1), \mathrm{dual}(H_2))$. For this note that for a suitable permutation π the following holds: $H_1 \cong H_2$ iff $\pi(H_1) \equiv H_2$ iff $\mathrm{dual}(\pi(H_1)) \equiv \mathrm{dual}(H_2)$

iff $\pi(\mathrm{dual}(H_1)) \equiv \mathrm{dual}(H_2)$ iff $\mathrm{dual}(H_1) \cong \mathrm{dual}(H_2)$. The rest of Statement 4 follows by the arguments used in Statement 3. □

Similar to problems related with one B-circuit we can check properties related with two B-circuits. For this we define:

$$\mathbf{E}(B) =_{\mathrm{def}} \{(C_1, C_2) \mid C_1 \text{ and } C_2 \text{ are } B\text{-circuits such that } f_{C_1} \text{ and } f_{C_2} \text{ have property } \mathbf{E}\}.$$

This gives the following proposition which is closely related with Proposition 2.19.

Proposition 4.7. *Let* $\mathbf{E}$ *be a property of Boolean functions, and let* B *and* B' *be finite sets of Boolean functions.*

1. *If* $B \subseteq \langle B' \rangle$ *then* $\mathbf{E}(B) \leq_{\mathrm{m}}^{\log} \mathbf{E}(B')$.
2. *If* $\langle B \rangle = \langle B' \rangle$ *then* $\mathbf{E}(B) \equiv_{\mathrm{m}}^{\log} \mathbf{E}(B')$.

Proof. Use the arguments given in proof of Proposition 2.19. □

4.3 Main results

In case $[B] \subseteq \mathrm{N}$ there exists a unique path from the output-gate to some input-gate or a gate which is labeled by a constant function. This is because every allowed Boolean function has one non-fictive variable at most. Hence the equivalence- and isomorphism-problem for such kinds of B-formulas and B-circuits is easy to solve. The next lemma will show this more precisely:

Lemma 4.8. *Let* B *be a finite set of Boolean functions and* $[B] \subseteq \mathrm{N}$. *Then* $\mathrm{EQ}^{\mathrm{C}}(B) \in \mathbf{L}$, $\mathrm{EQ}^{\mathrm{F}}(B) \in \mathbf{L}$, $\mathrm{ISO}^{\mathrm{C}}(B) \in \mathbf{L}$ *and* $\mathrm{ISO}^{\mathrm{F}}(B) \in \mathbf{L}$.

Proof. To show $\mathrm{EQ}^{\mathrm{C}}(B) \in \mathbf{L}$ and $\mathrm{ISO}^{\mathrm{C}}(B) \in \mathbf{L}$ note that all functions in B have at most 1 non-fictive variable. Therefore there exists a unique path from the output-gate to a 'target-gate', which is either an input-gate or a gate computing a constant. Let s_1 (s_2, resp.) be the output gate of C_1 (C_2, resp.) and t_1 (t_2, resp.) the 'target'-gate in C_1 (C_2, resp.).

For the first part of the statement we argue that $(C_1, C_2) \in \mathrm{EQ}^{\mathrm{C}}(B)$ iff either t_1 and t_2 are marked by the same variable and the number modulo 2 of non-gates in the paths is the same or t_1 and t_2 compute the same (different, resp.) constant function and the number modulo 2 of non-gates in the two paths is the same (is different, resp.). This can be checked in logarithmic space. Moreover we know by Proposition 4.6.1 that $\mathrm{EQ}^{\mathrm{F}}(B) \leq_{\mathrm{m}}^{\log} \mathrm{EQ}^{\mathrm{C}}(B)$ holds, showing that $\mathrm{EQ}^{\mathrm{F}}(B) \in \mathbf{L}$.

For the second part we mention that $(C_1, C_2) \in \mathrm{ISO}^{\mathrm{C}}(B)$ iff either t_1 and t_2 are marked by a variable and the number modulo 2 of non-gates in the paths is the same or both are marked by a constant and $C_1 \equiv C_2$. Again this can be checked in logarithmic space as seen above. The statements $\mathrm{ISO}^{\mathrm{F}}(B) \in \mathbf{L}$

immediately follows by Proposition 4.6.1 and the fact that **L** is closed under log-space reductions. □

If we restrict ourselves to vel-functions (et-functions, resp.) we will prove that the isomorphism- and equivalence-problem for such B-circuits is complete for **NL**. For this we use the graph accessibility problem (GAP), which is well known to be **NL**-complete. If we only use exclusive-or functions for our B-circuits we will show that the equivalence- and isomorphism-problem is $\oplus$**L**-complete. To show this we use the $\oplus$**L**-complete graph odd accessibility problem (GOAP) as a starting point.
In contrast to these results we will later show that the corresponding problems for B-formulas can be decided in deterministic logarithmic space.

Theorem 4.9. *Let B be a finite set of Boolean functions. If $\mathrm{E}_2 \subseteq [B] \subseteq \mathrm{E}$ or $\mathrm{V}_2 \subseteq [B] \subseteq \mathrm{V}$, then $\mathrm{EQ}^{\mathrm{C}}(B)$ and $\mathrm{ISO}^{\mathrm{C}}(B)$ are $\leq_{\mathrm{m}}^{\log}$-complete for* **NL**. *If $\mathrm{L}_2 \subseteq [B] \subseteq \mathrm{L}$, then $\mathrm{EQ}^{\mathrm{C}}(B)$ and $\mathrm{ISO}^{\mathrm{C}}(B)$ are $\leq_{\mathrm{m}}^{\log}$-complete for $\oplus$***L**.

Proof. First let $\mathrm{V}_2 \subseteq [B] \subseteq \mathrm{V}$.
Upper bound: By Theorem 3.4 we know that $\mathrm{VAL}^{\mathrm{C}}(B) \in$ **NL**. Let $C_1(x_1, \ldots, x_n)$ and $C_2(x_1, \ldots, x_n)$ be two B-circuits. The Boolean functions described by C_1 and C_2 can be expressed as follows: $f_{C_1}(x_1, \ldots, x_n) = a_0 \vee (a_1 \wedge x_1) \vee \cdots \vee (a_n \wedge x_n)$ and $f_{C_2}(x_1, \ldots, x_n) = b_0 \vee (b_1 \wedge x_1) \vee \cdots \vee (b_n \wedge x_n)$, where $a_1, \ldots, a_n, b_1, \ldots, b_n \in \{0, 1\}$.
Clearly $(C_1, C_2) \in \mathrm{EQ}^{\mathrm{C}}(B)$ iff either $a_0 = b_0 = 1$ or $a_0 = b_0 = 0$ and $a_i = b_i$ for $1 \leq i \leq n$. The values of a_i and b_i, where $0 \leq i \leq n$, can be determined by using the following fact: $a_0 = 0$ ($b_0 = 0$, resp.) iff $f_{C_1}(0^n) = 0$ ($f_{C_2}(0^n) = 0$, resp.) and $a_i = 0$ ($b_i = 0$, resp.) for $1 \leq i \leq n$ iff $a_0 = 0$ ($b_0 = 0$, resp.) and $f_{C_1}(0^{i-1}10^{n-i}) = 0$ ($f_{C_2}(0^{i-1}10^{n-i}) = 0$, resp.). This can be checked in logarithmic space with the help of $\mathrm{VAL}^{\mathrm{C}}(B)$ as an oracle. Since $\mathrm{VAL}^{\mathrm{C}}(B) \in$ **NL** we conclude by using Proposition 2.4 that $\mathrm{EQ}^{\mathrm{C}}(B)$ is in $\mathbf{L}^{\mathbf{NL}} = \mathbf{NL}$.
Similarly $(C_1, C_2) \in \mathrm{ISO}^{\mathrm{C}}(B)$ iff either $a_0 = b_0 = 1$ or $|\{i \mid a_i = 1, 1 \leq i \leq n\}| = |\{i \mid b_i = 1, 1 \leq i \leq n\}|$ and $a_0 = b_0 = 0$. Since the values of a_i and b_i can be obtained by an **NL**-computation and the cardinality of $\{i \mid a_i = 1, 1 \leq i \leq n\}$ and $\{i \mid b_i = 1, 1 \leq i \leq n\}$ can be compared and stored in logarithmic space, we conclude that $\mathrm{ISO}^{\mathrm{C}}(B) \in \mathbf{L}^{\mathbf{NL}} = \mathbf{NL}$ by Proposition 2.4.

Lower bound: We show that $\overline{\mathrm{GAP}} \leq_{\mathrm{m}}^{\log} \mathrm{EQ}^{\mathrm{C}}(B)$ and $\overline{\mathrm{GAP}} \leq_{\mathrm{m}}^{\log} \mathrm{ISO}^{\mathrm{C}}(B)$ hold. According to Proposition 4.7 it is sufficient to show $\overline{\mathrm{GAP}} \leq_{\mathrm{m}}^{\log} \mathrm{EQ}^{\mathrm{C}}(\{\mathrm{vel}\})$ and $\overline{\mathrm{GAP}} \leq_{\mathrm{m}}^{\log} \mathrm{ISO}^{\mathrm{C}}(\{\mathrm{vel}\})$. Given a directed acyclic graph G whose uniquely numbered vertices have outdegree 0 or 2, a start vertex s and a target vertex t, having outdegree 0. Next construct two similar B-circuits C_1 and C_2 as follows: Every vertex having outdegree 2 becomes a vel-gate, the vertex s becomes the output-gate of C_1 and C_2. Every outdegree 0 vertex which is not the target vertex t becomes an input-gate marked by x_j, where j is the unique number of this outdegree 0 vertex. Now the target-vertex t is replaced by the small circuit $\mathrm{vel}(y, y')$ in C_1 and with y in C_2. Finally we add another input-gate y'' in C_1

and two another input-gates y' and y'' in C_2, which are not connected with other gates.
The variable y'' is a (syntactically) fictive variable in $C_1(x_{j_1},\dots,x_{j_m},y,y',y'')$ and both y' and y'' are (syntactically) fictive variables in $C_2(x_{j_1},\dots,x_{j_m},y,y',y'')$, where $(m+1)$ is the number of outdegree 0 vertices in G. Obviously, if there exists no path from s to t then $(C_1,C_2)\in \mathrm{EQ^C}(B)$ and therefore $(C_1,C_2)\in \mathrm{ISO^C}(B)$ by $\pi=\mathrm{id}$, since the variables y, y' and y'' are fictive in both C_1 and C_2. Next assume that there is a path between s and t. In this case y,y' are not fictive in C_1 and y is not fictive in C_2, but y'' is fictive in C_1 and y',y'' are fictive in C_2. Therefore the number of non-fictive variables in C_1 is bigger by one than the number of non-fictive variables in C_2 and we conclude that $\#_1(C_1)>\#_1(C_2)$. By Proposition 4.5 we know that $C_1\not\cong C_2$ and therefore $(C_1,C_2)\notin \mathrm{ISO^C}(B)$ and in particular $(C_1,C_2)\notin \mathrm{EQ^C}(B)$. Because the described transformation can be calculated in logarithmic space this establishes the reduction. By Theorem 2.5 the statement follows.

Now let $\mathrm{E}_2\subseteq[B]\subseteq\mathrm{E}$. In this case we have already shown that $\mathrm{EQ^C}(\mathrm{dual}(B))$ and $\mathrm{ISO^C}(\mathrm{dual}(B))$ are $\leq_{\mathrm{m}}^{\log}$-complete for $\mathbf{NL}$. Using Proposition 4.6 the second statement follows immediately.

Finally let $\mathrm{L}_2\subseteq[B]\subseteq\mathrm{L}$.
Upper bound: Let C_1 and C_2 be B-circuits. The Boolean functions described by the B-circuits $C_1(x_1,\dots,x_n)$ and $C_2(x_1,\dots,x_n)$ can be expressed as follows: $f_{C_1}(x_1,\dots,x_n)=a_0\oplus(a_1\wedge x_1)\oplus\cdots\oplus(a_n\wedge x_n)$ and $f_{C_2}(x_1,\dots,x_n)=b_0\oplus(b_1\wedge x_1)\oplus\cdots\oplus(b_n\wedge x_n)$, where $a_1,\dots,a_n,b_1,\dots,b_n\in\{0,1\}$.
Clearly $(C_1,C_2)\in\mathrm{EQ^C}(B)$ iff $a_i=b_i$ for $0\leq i\leq n$. Similar to the case $\mathrm{V}_2\subseteq[B]\subseteq\mathrm{V}$ the values a_i and b_i for $0\leq i\leq n$ can be determined by a $\oplus\mathbf{L}$-calculation, since we know by Theorem 3.4 that $\mathrm{VAL^C}(B)\in\oplus\mathbf{L}$. In particular $a_0=f_{C_1}(0,\dots,0)$, $b_0=f_{C_2}(0,\dots,0)$, $a_i=f_{C_1}(0^{i-1}10^{n-i})\oplus a_0$ and $b_i=f_{C_2}(0^{i-1}10^{n-i})\oplus b_0$, where $1\leq i\leq n$. Hence $\mathrm{EQ^C}(B)\in\mathbf{L}^{\oplus\mathbf{L}}$ and by Proposition 2.4 $\mathrm{EQ^C}(B)\in\oplus\mathbf{L}$.
For the upper bound of $\mathrm{ISO^C}(B)$ we argue that $(C_1,C_2)\in\mathrm{ISO^C}(B)$ iff $a_0=b_0$ and $|\{i\mid a_i=1,1\leq i\leq n\}|=|\{i\mid b_i=1,1\leq i\leq n\}|$. Because the values a_i and b_i can be obtained by a $\oplus\mathbf{L}$-computation we end up with $\mathrm{ISO^C}(B)\in\mathbf{L}^{\oplus\mathbf{L}}=\oplus\mathbf{L}$ by Proposition 2.4.

Lower bound: We show that $\overline{\mathrm{GOAP}}\leq_{\mathrm{m}}^{\log}\mathrm{EQ^C}(B)$ and $\overline{\mathrm{GOAP}}\leq_{\mathrm{m}}^{\log}\mathrm{ISO^C}(B)$ hold. Because of Proposition 4.7 it is sufficient to show that $\overline{\mathrm{GOAP}}\leq_{\mathrm{m}}^{\log}\mathrm{EQ^C}(\{h\})$ and $\overline{\mathrm{GOAP}}\leq_{\mathrm{m}}^{\log}\mathrm{ISO^C}(\{h\})$, where $h(x,y,z)=_{\mathrm{def}}x\oplus y\oplus z$ is a base of L_2 (cf. Theorem 2.11). Now construct a $\{h\}$-circuit C_1 as follows: Every outdegree 2 vertex becomes a h-gate, where an additional incoming edge is connected to a z input-gate. The target vertex is replaced by the small circuit $y\oplus y'\oplus y''$. Every other outdegree 0 vertex becomes a z input-gate. Clearly this circuit computes $g(z)$ for some suitable Boolean function g iff $G\in\overline{\mathrm{GOAP}}$ and $y\oplus y'\oplus y''\oplus g(z)$ otherwise. Secondly build a $\{h\}$-circuit C_2 in the same way, but replace the target vertex by a y-gate and add y',y'' as syntactically fictive variables. Hence C_2 computes $g(z)$

iff $G \in \overline{\mathrm{GOAP}}$ and $y \oplus g(z)$ otherwise. This gives $G \in \overline{\mathrm{GOAP}}$ iff C_1 and C_2 compute $g(z)$ iff $(C_1, C_2) \in \mathrm{EQ}^{\mathrm{C}}(\{h\})$. Additionally we know if $G \in \overline{\mathrm{GOAP}}$ then $(C_1, C_2) \in \mathrm{ISO}^{\mathrm{C}}(\{h\})$ via $\pi = \mathrm{id}$. Now let $G \notin \overline{\mathrm{GOAP}}$. In this case the number of non-fictive variables in C_1 is bigger than the number of non-fictive variables in C_2, showing that $C_1 \not\cong C_2$ and $\overline{\mathrm{GOAP}} \leq_{\mathrm{m}}^{\log} \mathrm{ISO}^{\mathrm{C}}(\{h\})$. □

Theorem 3.4 shows that in the cases $\mathrm{V}_2 \subseteq [B] \subseteq \mathrm{V}$ and $\mathrm{E}_2 \subseteq [B] \subseteq \mathrm{E}$ the circuit value problem is complete for **NL** and in the case $\mathrm{L}_2 \subseteq [B] \subseteq \mathrm{L}$ that it is complete for $\oplus\mathbf{L}$. The next two lemmas show that this does not hold for B-formulas as well. In contrast to the circuit value problem the formula value problem $\mathrm{VAL}^{\mathrm{F}}(B)$ is in **L** in all these three cases, which gives us a hint that the equality- and isomorphism-problem for B-formulas is easier in this cases.

Lemma 4.10. *Let B be a finite set of Boolean functions. Then the following the statements hold.*

1. *If* $\mathrm{V}_2 \subseteq [B] \subseteq \mathrm{V}$ *then* $\mathrm{VAL}^{\mathrm{F}}(B) \leq_{\mathrm{m}}^{\log} \mathrm{VAL}^{\mathrm{F}}(\{\mathrm{vel}\})$.
2. *If* $\mathrm{E}_2 \subseteq [B] \subseteq \mathrm{E}$ *then* $\mathrm{VAL}^{\mathrm{F}}(B) \leq_{\mathrm{m}}^{\log} \mathrm{VAL}^{\mathrm{F}}(\{\mathrm{et}\})$.
3. *If* $\mathrm{L}_2 \subseteq [B] \subseteq \mathrm{L}$ *then* $\mathrm{VAL}^{\mathrm{F}}(B) \leq_{\mathrm{m}}^{\log} \mathrm{VAL}^{\mathrm{F}}(\{\mathrm{aut}\})$.

Proof. First let $\mathrm{V}_2 \subseteq [B] \subseteq \mathrm{V}$. Note that any function $f \in B$ can be represented as

$$\begin{aligned} f(x_1, \ldots, x_m) &= a_0 \vee (a_1 \wedge x_1) \vee \cdots \vee (a_m \wedge x_m) \\ &= a_0 \vee \bigvee_{a_i=1} x_i, \text{ where } a_0, \ldots, a_m \in \{0,1\}. \end{aligned}$$

Let (H, α) be an instance of $\mathrm{VAL}^{\mathrm{F}}(B)$. We construct a $\{\mathrm{vel}, 0, 1\}$-formula H' as follows: Every function-symbol R in the B-formula H is replaced by a $\{\mathrm{vel}, 0, 1\}$-formula R', which represents the same Boolean function. If the function-symbol R has fictive inputs, the edges connected with these inputs will be deleted in H'. Since each variable of R occurs only once in R', the size of H' is polynomially bounded in the size of H. Therefore $(H, \alpha) \in \mathrm{VAL}^{\mathrm{F}}(B)$ iff $(H', \alpha) \in \mathrm{VAL}^{\mathrm{F}}(\{\mathrm{vel}, 0, 1\})$ and $\mathrm{VAL}^{\mathrm{F}}(B) \leq_{\mathrm{m}}^{\log} \mathrm{VAL}^{\mathrm{F}}(\{\mathrm{vel}, 0, 1\})$, because the described transformation can be calculated in logarithmic space (note that B is finite).
Now replace all occurrences of 0 (1, resp.) by a new variable, call the resulting formula H'' and construct a new assignment α' by assigning 0 (1, resp.) to all these variables. Clearly $(H', \alpha) \in \mathrm{VAL}^{\mathrm{F}}(\{\mathrm{vel}, 0, 1\})$ iff $(H'', \alpha') \in \mathrm{VAL}^{\mathrm{F}}(\{\mathrm{vel}\})$, showing that $\mathrm{VAL}^{\mathrm{F}}(B) \leq_{\mathrm{m}}^{\log} \mathrm{VAL}^{\mathrm{F}}(\{\mathrm{vel}\})$.

Let $\mathrm{E}_2 \subseteq [B] \subseteq \mathrm{E}$ ($\mathrm{L}_2 \subseteq [B] \subseteq \mathrm{L}$, resp.). By using the same technique as above, the fact that any function $f \in B$ can be represented as $f(x_1, \ldots, x_m) = a_0 \wedge (a_1 \vee x_1) \wedge \cdots \wedge (a_m \vee x_m)$ ($f(x_1, \ldots, x_m) = a_0 \oplus (a_1 \wedge x_1) \oplus \cdots \oplus (a_m \wedge x_m)$, resp.) and replacing all symbols for vel by the symbol for et (aut, resp.) the result follows. □

Lemma 4.11. *Let B be a finite set of Boolean functions such that* $\mathrm{V}_2 \subseteq [B] \subseteq \mathrm{V}$, $\mathrm{E}_2 \subseteq [B] \subseteq \mathrm{E}$ *or* $\mathrm{L}_2 \subseteq [B] \subseteq \mathrm{L}$, *then* $\mathrm{VAL}^{\mathrm{F}}(B) \in \mathbf{L}$.

Proof. For the first case let B be a finite set of Boolean functions such that $\mathrm{V}_2 \subseteq [B] \subseteq \mathrm{V}$ holds. As shown in Lemma 4.10 it is sufficient to restrict ourselves to $\{\mathrm{vel}\}$-formulas. Since B-formulas are defined to be B-circuits having a fan-out of at most 1, they form a tree, where the leafs are the input-variables and the root is the output-gate t. Clearly we can check in logarithmic space, whether for a given gate s there is a non-fictive path from s to t. Now let $H(x_1, \ldots, x_n)$ be a $\{\mathrm{vel}\}$-formula and $\alpha = (a_1, \ldots, a_n) \in \{0,1\}^n$ be an assignment for H. Since $(H, \alpha) \in \mathrm{VAL}^{\mathrm{F}}(B)$ iff there exists an input-variable x_i, where $1 \leq i \leq n$, such that $a_i = 1$ and there exists a non-fictive path from x_i to t, we can check $(H, \alpha) \in \mathrm{VAL}^{\mathrm{F}}(B)$ in deterministic logarithmic space.

The second case $\mathrm{E}_2 \subseteq [B] \subseteq \mathrm{E}$ can be shown similarly to the previous case. For this observe that $(H, \alpha) \notin \mathrm{VAL}^{\mathrm{F}}(B)$ iff there exists an input-gate x_i, where $1 \leq i \leq n$, such that $\alpha_i = 0$ and there exists a non-fictive path for x_i to t. Again this can be checked in logarithmic space. The statement follows, since deterministic logarithmic space is obviously closed under complement.

For the third case let B be a finite set of Boolean functions such that $\mathrm{L}_2 \subseteq [B] \subseteq \mathrm{L}$. As shown in Lemma 4.10 it is sufficient to use $\mathrm{VAL}^{\mathrm{F}}(\{\mathrm{aut}\})$ as an upper bound. Obviously $(H, \alpha) \in \mathrm{VAL}^{\mathrm{F}}(\{\mathrm{aut}\})$ iff the number of input-variables x_i, where $1 \leq i \leq n$, such that $a_i = 1$ and there is a non-fictive path which starts at x_i and ends at t, is odd. (Note that more than one input-gate can be labeled with the same variable x_i, since all gates have a fan-out of at most 1. Hence it can be the case that we have to consider the same variable more than once.) Again this can be checked in logarithmic space, showing the third statement. □

The next theorem clarifies the succinctness of circuits. When $\mathrm{EQ}^{\mathrm{C}}(B)$ and $\mathrm{ISO}^{\mathrm{C}}(B)$ are $\mathbf{NL}$-complete or $\oplus\mathbf{L}$-complete, the formula case is still easy to solve, as we will see in the Theorem below.

Theorem 4.12. *Let B be a finite set of Boolean functions. If $\mathrm{E}_2 \subseteq [B] \subseteq \mathrm{E}$, $\mathrm{V}_2 \subseteq [B] \subseteq \mathrm{V}$ or $\mathrm{L}_2 \subseteq [B] \subseteq \mathrm{L}$, then $\mathrm{EQ}^{\mathrm{F}}(B) \in \mathbf{L}$ and $\mathrm{ISO}^{\mathrm{F}}(B) \in \mathbf{L}$.*

Proof. In the proof of Theorem 4.9 we showed that $\mathrm{EQ}^{\mathrm{C}}(B) \in \mathbf{L}^{\mathrm{VAL}^{\mathrm{C}}(B)}$ and $\mathrm{ISO}^{\mathrm{C}}(B) \in \mathbf{L}^{\mathrm{VAL}^{\mathrm{C}}(B)}$. By the same construction we see that $\mathrm{EQ}^{\mathrm{F}}(B) \in \mathbf{L}^{\mathrm{VAL}^{\mathrm{F}}(B)}$ and $\mathrm{ISO}^{\mathrm{F}}(B) \in \mathbf{L}^{\mathrm{ISO}^{\mathrm{F}}(B)}$. By using Proposition 2.4 and Lemma 4.11 the statement follows. □

It is well known that $\mathbf{NC^1} \subseteq \mathbf{L}$ (see [Vol99]), where $\mathbf{NC^1}$ is the class of problems which can be solved by $\{\mathrm{et}, \mathrm{vel}, \mathrm{non}\}$-circuits of polynomial size (i.e., the number of gates), bounded fan-in and logarithmic depth (i.e., the length of the longest path from an input-gate to the output-gate). Moreover Buss showed in [Bus87], that $\mathrm{VAL}^{\mathrm{F}}(\{\mathrm{et}, \mathrm{vel}, \mathrm{non}\})$ is complete for $\mathbf{NC^1}$. Hence $\mathrm{VAL}^{\mathrm{F}}(\{\mathrm{vel}\}) \in \mathbf{NC^1}$ and $\mathrm{VAL}^{\mathrm{F}}(\{\mathrm{et}\}) \in \mathbf{NC^1}$. If $\mathrm{VAL}^{\mathrm{F}}(\{\mathrm{aut}\}) \in \mathbf{NC^1}$ holds too, then Lemma 4.12 can be refined to $\mathrm{EQ}^{\mathrm{F}}(B) \in \mathbf{L}^{\mathbf{NC^1}}$ and $\mathrm{ISO}^{\mathrm{F}}(B) \in \mathbf{L}^{\mathbf{NC^1}}$.
Moreover the result for $\mathrm{EQ}^{\mathrm{F}}(\{\mathrm{vel}\})$ can be easily improved by the following observation: The $\mathbf{L}$ base-computation used in the proof of Theorem 4.12 can be replaced by an $\mathbf{AC^0}$ computation, where $\mathbf{AC^0}$ is the class of problems which

can be solved by $\{\mathrm{vel}, \mathrm{et}, \mathrm{non}\}$-circuits of polynomial size, unbounded fan-in and constant depth, because two $\{\mathrm{vel}\}$-circuits C_1 and C_2, such that $f_{C_1}(x_1, \ldots, x_n) = a_0 \vee (a_1 \wedge x_1) \vee \cdots \vee (a_n \wedge x_n)$ and $f_{C_2}(x_1, \ldots, x_n) = b_0 \vee (b_1 \wedge x_1) \vee \cdots \vee (b_n \wedge x_n)$ are equal (i.e., $C_1 \equiv C_2$) iff $(a_0 \leftrightarrow 1 \wedge b_0 \leftrightarrow 1) \oplus (a_0 \leftrightarrow 0 \wedge b_0 \leftrightarrow 0 \wedge \bigwedge_{i=1}^{n} a_i \leftrightarrow b_i)$ is satisfied. Hence $\mathrm{EQ}^{\mathrm{F}}(\{\mathrm{vel}\})$ can be solved by an $\mathbf{AC^0}$-circuit equipped with $\mathrm{VAL}^{\mathrm{F}}(\{\mathrm{vel}\})$ oracle-gates. This shows that $\mathrm{EQ}^{\mathrm{F}}(\{\mathrm{vel}\}) \in (\mathbf{AC^0})^{\mathrm{VAL}^{\mathrm{F}}(\{\mathrm{vel}\})} \subseteq (\mathbf{AC^0})^{\mathbf{NC^1}} \subseteq (\mathbf{NC^1})^{\mathbf{NC^1}} = \mathbf{NC^1}$ (for the last equality see [Vol99], Theorem 4.21). A similar result can be achieved for $\mathrm{EQ}^{\mathrm{F}}(\{\mathrm{et}\})$.

The next lemma shows that it is possible to construct for every monotone 3-DNF-formula an equivalent $(B \cup \{0,1\})$-formula in logarithmic space, if $(B \cup \{0,1\})$ is a base for M. This means that we are able to avoid an exponential blow-up of the size of the resulting $(B \cup \{0,1\})$-formula.

Lemma 4.13. *Let $k > 0$ be fixed and B be a finite set of Boolean functions, such that $E(x, y, v, u)$ and $V(x, y, v, u)$ are B-formulas, fulfilling $E(x, y, 0, 1) \equiv x \wedge y$ and $V(x, y, 0, 1) \equiv x \vee y$. Then, for any monotone k-DNF formula $H(x_1, \ldots, x_n)$, there exists a B-formula $H'(x_1, \ldots, x_n, u, v)$ such that $H'(x_1, \ldots, x_n, 0, 1) \equiv H(x_1, \ldots, x_n)$. Moreover, H' can be computed from H in logarithmic space.*

Proof. Since $\{\wedge, \vee\}$ forms a basis for all monotone functions (without constants), we can trivially transform every monotone k-DNF formula into a $(B \cup \{0,1\})$-formula. In the case of a given monotone k-DNF formula with m clauses, to avoid an exponential blow-up while replacing the m $\vee$'s, we insert parentheses in such a way that we get a tree of depth $\log m$, and then replace each $\vee$ by an appropriate $(B \cup \{0,1\})$-formula. Finally, the $k-1$ $\wedge$'s inside the clauses are replaced by suitable B-formulas. □

In the following we show how to build two monotone formulas out of a 3-DNF formula, such that these two monotone formulas are equivalent iff the 3-DNF formula is a tautology.

Lemma 4.14. *Let B be a finite set of Boolean functions such that $\{\mathrm{vel}, \mathrm{et}\} \subseteq [B]$. Then there exist logspace computable B-formulas H_1 and H_2, which can be computed out of H, such that $H \in$ 3-TAUT iff $H_1 \equiv H_2$ iff $H_1 \cong H_2$ iff $\#_1(H_1) = \#_1(H_2)$.*

Moreover the formulas H_1 and H_2 do not represent constant Boolean functions (i.e., $H_1 \not\equiv 0$, $H_1 \not\equiv 1$, $H_2 \not\equiv 0$ and $H_1 \not\equiv 1$).

Proof. In the following we describe the needed logspace computable function. For this let $H(x_1, \ldots, x_n)$ be a 3-DNF formula (remember that all instances of 3-TAUT are in 3-DNF) and define $V_- =_{\mathrm{def}} \{x_{j_1}, x_{j_2}, \ldots, x_{j_m}\}$ the set of variables that occur negated in H. If $V_- \neq \emptyset$, then let $H'(x_1, \ldots, x_n, y_{j_1}, y_{j_2}, \ldots, y_{j_m})$ be the monotone formula that emerges when we replace all negated variables by new unnegated variables $\{y_{j_1}, y_{j_2}, \ldots, y_{j_m}\}$ in H and define the monotone formula H'' as follows: $H''(x_1, \ldots, x_n, y_{j_1}, y_{j_2}, \ldots, y_{j_m}) =_{\mathrm{def}} \bigwedge_{l=1}^{m}(x_{j_l} \vee y_{j_l})$. By Lemma 4.13 we can construct B-formulas H_1 and H_2 in logarithmic space, such that $H_1 \equiv H' \wedge H''$ and $H_2 \equiv H''$. In the case that $V_- = \emptyset$ let $H_1 \equiv x \wedge y$ and $H_2 \equiv x \vee y$.

First let $V_- = \emptyset$. Since any monotone 3-DNF formula $\Phi(x_1, \ldots, x_n)$ (note that we are not allowed to use constants) is not a tautology, because Φ is clearly not satisfied by the assignment $\{x_1 := 0, x_2 := 0, \ldots, x_n := 0\}$, we give $x \wedge y$ and $x \vee y$ as the output. This fulfills the statement, since $x \wedge y \not\equiv x \vee y$, $x \wedge y \not\cong x \vee y$ and $\#_1(x \wedge y) \neq \#_1(x \vee y)$.

For the following claim let $V_- \neq \emptyset$:
Claim: $H \in$ 3-TAUT iff $H_1 \equiv H_2$ iff $H_1 \cong H_2$ iff $\#_1(H_1) = \#_1(H_2)$.
"$\Rightarrow$": Let I be an arbitrary assignment which is compatible with H_1 and H_2. If there exists an i such that $I/\{x_{j_i}\} = \{x_{j_i} := 0\}$ and $I/\{y_{j_i}\} = \{y_{j_i} := 0\}$ then $I \not\models H_1$ and $I \not\models H_2$, where $1 \leq i \leq n$. In the other case we know that for all $I/\{x_{j_i}\} = \{x_{j_i} := a_{j_i}\}$ and $I/\{y_{j_i}\} = \{y_{j_i} := b_{j_i}\}$ $a_{j_i} = 1$ or $b_{j_i} = 1$ holds. Clearly $I \models H'$ since $H \in$ 3-TAUT and we simply replaced all negated variables H by positive variables. To be more precise, remember that all assignments I, having the property that for all $x_{j_i} \in V_-$ either $I/\{x_{j_i}\} = \{x_{j_i} := 1\}$ and $I/\{y_{j_i}\} = \{y_{j_i} := 0\}$ or $I/\{x_{j_i}\} = \{x_{j_i} := 0\}$ and $I/\{y_{j_i}\} = \{y_{j_i} := 1\}$ holds, simulate an assignment $I' =_{\text{def}} I/\{x_1, \ldots, x_n\}$ on H correctly. In the case that there exists an $1 \leq i \leq m$ such that $I/\{x_{j_i}\} = \{x_{j_i} := 1\}$ and $I/\{y_{j_i}\} = \{y_{j_i} := 1\}$ we conclude that $I \models H'$ more than ever, because we have replaced any negated variable by a positive one. Moreover $I \models H''$, giving us that $I \models H_1$ and $I \models H_2$. Hence $H_1 \equiv H_2$, $\#_1(H_1) = \#_1(H_2)$ and $H_1 \cong H_2$ with $\pi = \text{id}$.
"$\Leftarrow$": Let $H(x_1, \ldots, x_n) \notin$ 3-TAUT, then there exists an assignment I' such that $I' \not\models H$ and $I'/\{x_{j_1}, x_{j_2}, \ldots, x_{j_m}\} = \{x_{j_1} := a_{j_1}, x_{j_2} := a_{j_2}, \ldots, x_{j_m} := a_{j_m}\}$. We define I as follows: $I =_{\text{def}} I' \cup \{y_{j_1} := 1 - a_{j_1}, y_{j_2} := 1 - a_{j_2}, \ldots, y_{j_m} := 1 - a_{j_m}\}$. Clearly $I \not\models H'$, because for all $x_{j_i} \in V_-$ we know that either $I/\{x_{j_i}\} = \{x_{j_i} := 1\}$ and $I/\{y_{j_i}\} = \{y_{j_i} := 0\}$ or $I/\{x_{j_i}\} = \{x_{j_i} := 0\}$ and $I/\{y_{j_i}\} = \{y_{j_i} := 1\}$, which simulates the original assignment I' on H. Hence $I \not\models H_1$, but $I \models H_2$, which shows that $H_1 \not\equiv H_2$ and $\#_1(H_1) < \#_1(H_2)$. To show this note that if $I \models H_1$ then $I \models H_2$. Hence $\#_1(H_1) \leq \#_1(H_2)$. Now assume that $\#_1(H_1) = \#_1(H_2)$ and therefore $\#_0(H_1) = \#_0(H_2)$ hold. We know that there is an assignment $I \not\models H_1$, but $I \models H_2$. Because of this there must exists an assignment I'' such that $I'' \models H_1$, but $I'' \not\models H_2$, because $\#_0(H_1) = \#_0(H_2)$. This is a contradiction, showing that $\#_1(H_2) \neq \#_1(H_1)$ and by Proposition 4.5 we conclude that $H_1 \not\cong H_2$.

Finally we summarize $H \in$ 3-TAUT iff $H_1 \equiv H_2$, $H \in$ 3-TAUT iff $\#_1(H_2) = \#_1(H_1)$ and $H \in$ 3-TAUT iff $H_1 \cong H_2$. Additionally note that $(0, \ldots, 0) \not\models H_1$ and $(0, \ldots, 0) \not\models H_2$, but $(1, \ldots, 1) \models H_1$ and $(1, \ldots, 1) \models H_2$. Hence the formulas H_1 and H_2 clearly can not represent a constant function. $\square$

The property that the formulas H_1 and H_2 can not represent a constant Boolean function will play an important role in the proof of Theorem 4.17. Moreover we will make use of the special form of H_1 and H_2 there.

Theorem 4.15. *Let B be a finite set of Boolean functions, such that* $\mathrm{et} \in [B]$ *and* $\mathrm{vel} \in [B \cup \{1\}]$. *Then* $\mathrm{EQ^F}(B)$ *is* $\leq_{\mathrm{m}}^{\log}$*-complete for* **coNP** *and* $\mathrm{ISO^F}(B)$ *is* $\leq_{\mathrm{m}}^{\log}$*-hard for* **coNP**.

Proof. Upper bound for $\mathrm{EQ^F}(B)$ is given by Proposition 4.6.2.
Lower bound: Since $\mathrm{et} \in [B]$ and $\mathrm{vel} \in [B \cup \{1\}]$ there exists B-formulas $E(x,y)$ and $V(x,y,z)$, such that $E(x,y) \equiv x \wedge y$ and $V(x,y,1) \equiv x \vee y$. Hence by Lemma 4.14 there exists logspace computable B-formulas $H_1(x_1,\ldots, x_n, y_{j_1},\ldots y_{j_m}, u)$ and $H_2(x_1,\ldots,x_n,y_{j_1},\ldots y_{j_m},u)$ such that $H \in$ 3-TAUT iff $H_1(x_1,\ldots,x_n,y_{j_1},\ldots y_{j_m},1) \equiv H_2(x_1,\ldots,x_n,y_{j_1},\ldots y_{j_m},1)$ and $H \in$ 3-TAUT iff $H_1(x_1,\ldots,x_n,y_{j_1},\ldots y_{j_m},1) \cong H_2(x_1,\ldots,x_n,y_{j_1},\ldots y_{j_m},1)$.

Claim 1: $H \in$ 3-TAUT iff $(H_1 \wedge u, H_2 \wedge u) \in \mathrm{EQ^F}(B)$.
For all assignments I such that $I/\{u\} = \{u := 0\}$ it holds that $I \not\models H_1 \wedge u$ and $I \not\models H_2 \wedge u$. For the next argument let I be an assignment which has the property $I/\{u\} = \{u := 1\}$. Clearly $H_1(x_1,\ldots,x_n,y_{j_1},\ldots,y_{j_m},1) \wedge 1 \equiv H_1(x_1,\ldots,x_n,y_{j_1},\ldots,y_{j_m},1)$ and $H_2(x_1,\ldots,x_n,y_{j_m},\ldots,y_{j_l},1) \wedge 1 \equiv H_2(x_1, \ldots,x_n,y_{j_m},\ldots,y_{j_l},1)$, which shows that $H \in$ 3-TAUT iff $(H_1 \wedge u, H_2 \wedge u) \in \mathrm{EQ^F}(B)$ by Lemma 4.14.
Claim 2: $H \in$ 3-TAUT iff $(H_1 \wedge u, H_2 \wedge u) \in \mathrm{ISO^F}(B)$.
By Claim 1 we know that if $H \in$ 3-TAUT then $H_1 \wedge u \equiv H_2 \wedge u$ and therefore $H_1 \wedge u \cong H_2 \wedge u$ via $\pi = \mathrm{id}$. Conversely if $H \notin$ 3-TAUT, then by Lemma 4.14 $\#_1(H_1(x_1,\ldots,x_n,y_{j_1},\ldots,y_{j_m},u) \wedge u) = \#_1(H_1(x_1,\ldots,x_n,y_{j_1},\ldots,y_{j_m},1)) \neq \#_1(H_2(x_1,\ldots,x_n,y_{j_1},\ldots,y_{j_m},1)) = \#_1(H_2(x_1,\ldots,x_n,y_{j_1},\ldots,y_{j_m},u) \wedge u)$ and by using Proposition 4.5 we conclude $H_1 \wedge u \ncong H_2 \wedge u$.

Because this reduction can be calculated in logarithmic space we deduce that 3-TAUT $\leq_{\mathrm{m}}^{\log}$ $\mathrm{EQ^F}(B)$ and 3-TAUT $\leq_{\mathrm{m}}^{\log}$ $\mathrm{ISO^F}(B)$. The statement follows by Proposition 2.6. □

Corollary 4.16. *Let B be a set of Boolean functions such that* $\mathrm{S}_{10} \subseteq [B]$ *or* $\mathrm{S}_{00} \subseteq [B]$. *Then* $\mathrm{EQ^F}(B)$ *and* $\mathrm{EQ^C}(B)$ *are* $\leq_{\mathrm{m}}^{\log}$*-complete for* **coNP** *and* $\mathrm{ISO^F}(B)$ *and* $\mathrm{ISO^C}(B)$ *are* $\leq_{\mathrm{m}}^{\log}$*-hard for* **coNP**.

Proof. First let $\mathrm{S}_{10} \subseteq [B]$. By Theorem 2.11 we know that $g(x,y,z) = x \wedge (y \vee z)$ is a base of S_{10}. Since $g(x,y,y) = x \wedge y$ and $g(1,x,y) = x \vee y$ we know that $\mathrm{et} \in [B]$ and $\mathrm{vel} \in [B \cup \{1\}]$. By Theorem 4.15 we conclude that $\mathrm{EQ^F}(B)$ is $\leq_{\mathrm{m}}^{\log}$-complete for **coNP** and $\mathrm{ISO^F}(B)$ is $\leq_{\mathrm{m}}^{\log}$-hard for **coNP**. Now let $\mathrm{S}_{00} \subseteq [B]$. By inspecting Figure 2.2 we obtain that $\mathrm{S}_{10} \subseteq [B]$ as well, or $\mathrm{S}_{10} \subseteq \mathrm{dual}([B])$. Use Proposition 4.6.3 and 4.6.4 to complete the proof for $\mathrm{EQ^F}(B)$ and $\mathrm{ISO^F}(B)$. Finally, by Proposition 4.6.1 and 4.6.2 the circuit case follows immediately. □

Now only two closed classes of Boolean functions are left: D and D_2. In this case the construction of Theorem 4.15 cannot work, since we see by Figure 2.2 that neither $\mathrm{et} \in \mathrm{D}_2$ nor $\mathrm{vel} \in \mathrm{D}_2$ (note that $\mathrm{E}_2 \nsubseteq \mathrm{D}_2$ and $\mathrm{V}_2 \nsubseteq \mathrm{D}_2$). Hence we have no possibility to use the et-function to force an additional variable to 1 for all satisfying assignments. In the following we will see that this is not needed.

Instead we use two new variables as a replacement for 0 and 1 and show, that in any case we get either two formulas representing the same constant function or formulas which match to Lemma 4.14.

Theorem 4.17. *Let B be a finite set of Boolean functions such that $\mathrm{D}_2 \subseteq [B] \subseteq \mathrm{D}$. Then $\mathrm{EQ}^{\mathrm{F}}(B)$ and $\mathrm{EQ}^{\mathrm{C}}(B)$ are $\leq_{\mathrm{m}}^{\log}$-complete for* **coNP** *and $\mathrm{ISO}^{\mathrm{F}}(B)$ and $\mathrm{ISO}^{\mathrm{C}}(B)$ are $\leq_{\mathrm{m}}^{\log}$-hard for* **coNP**.

Proof. Upper bound for $\mathrm{EQ}^{\mathrm{F}}(B)$ and $\mathrm{EQ}^{\mathrm{C}}(B)$ is given by Proposition 4.6.2.
Lower bound for $\mathrm{EQ}^{\mathrm{C}}(B)$. By Theorem 2.11 we know that $g(x,y,z) = (x \wedge y) \vee (x \wedge z) \vee (y \wedge z)$ is a base for D_2. Clearly $g(x,y,0) = x \wedge y$ and $g(x,y,1) = x \vee y$, showing that $\mathrm{et}, \mathrm{vel} \in [B \cup \{0,1\}]$. Hence there exists B-formulas $E(x,y,z)$ and $V(x,y,z)$ (not necessarily different) such that $E(x,y,0) \equiv V(x,y,0) \equiv x \wedge y$ and $V(x,y,1) \equiv E(x,y,1) \equiv x \vee y$.

By using Lemma 4.14 we can build two B-formulas $H_1(x_1,\ldots,x_n,y_{j_1},y_{j_2},\ldots,y_{j_m},u,v)$ and $H_2(x_1,\ldots,x_n,y_{j_1},y_{j_2},\ldots,y_{j_m},u,v)$, where u is the replacement for the constant 0 and v is the replacement for 1, such that $H(x_1,\ldots,x_n) \in 3\text{-TAUT}$ iff $H_1(x_1,\ldots,x_n,y_{j_1},y_{j_2},\ldots,y_{j_m},0,1) \equiv H_2(x_1,\ldots,x_n,y_{j_1},y_{j_2},\ldots,y_{j_m},0,1)$.

Since the transformation of H into H_1 and H_2 can be calculated in logarithmic space, we only have to prove that

$$H \in 3\text{-TAUT iff } (V(u,E(H_1,v,u),v), V(u,E(H_2,v,u),v)) \in \mathrm{EQ}^{\mathrm{F}}(B).$$

Note that it is possible that a variable in V or E occurs more than one time. Despite that possibility, Lemma 4.14 takes care that the length of H_1 and H_2 is polynomially bounded in the length of H and $V(u,E(H_1,v,u),v)$, $V(u,E(H_2,v,u),v))$ can be calculated in logarithmic space.

In the following let I be an arbitrary assignment compatible with H_1 and H_2.

Case 1: $I/\{u,v\} = \{u := 0, v := 0\}$.
In this case clearly $V(x,y,0) \equiv E(x,y,0) \equiv x \wedge y$, $V(u,E(H_1(x_1,\ldots,x_n,y_{j_1},y_{j_2},\ldots,y_{j_m},u,v),v,u),v) \equiv u \wedge (H_1' \wedge v) \equiv 0$ and $V(u,E(H_2(x_1,\ldots,x_n,y_{j_1},y_{j_2},\ldots,y_{j_m},u,v),v,u),v) \equiv u \wedge (H_2' \wedge v) \equiv 0$, where H_1' and H_2' are the formulas that emerge out of H_1 and H_2 by substituting u and v by 0. Therefore $I \not\models V(u,E(H_1,v,u),v)$ and $I \not\models V(u,E(H_2,v,u),v)$.

Case 2: $I/\{u,v\} = \{u := 1, v := 1\}$.
Similarly to Case 1 we know that $V(x,y,1) \equiv E(x,y,1) \equiv x \vee y$. Hence $V(u,E(H_1(x_1,\ldots,x_n,y_{j_1},y_{j_2},\ldots,y_{j_m},u,v),v,u),v) \equiv u \vee (H_1' \vee v) \equiv 1$ and $V(u,E(H_2(x_1,\ldots,x_n,y_{j_1},y_{j_2},\ldots,y_{j_m},u,v),v,u),v) \equiv u \vee (H_2' \vee v) \equiv 1$, where H_1' and H_2' are the formulas that emerge out of H_1 and H_2 by substituting u and v by 1. Therefore $I \models V(u,E(H_1,v,u),v)$ and $I \models V(u,E(H_2,v,u),v)$.

Case 3: $I/\{u,v\} = \{u := 0, v := 1\}$.
In this case we have $V(u,E(H_1,v,u),v) \equiv u \vee (H_1 \wedge v) \equiv H_1(x_1,\ldots,x_n,y_{j_1},y_{j_2},\ldots,y_{j_m},0,1)$, $V(u,E(H_2,v,u),v) \equiv u \vee (H_2 \wedge v) \equiv H_2(x_1,\ldots,x_n,y_{j_1},y_{j_2},\ldots,y_{j_m},0,1)$ and by Lemma 4.14 we know that $H \in 3\text{-TAUT}$ iff $H_1(x_1,\ldots,x_n,y_{j_1},y_{j_2},\ldots,y_{j_m},0,1) \equiv H_2(x_1,\ldots,x_n,y_{j_1},y_{j_2},\ldots,y_{j_m},0,1)$.

Hence we conclude $H \in$ 3-TAUT iff $V(u, E(H_1, v, u), v) \equiv V(u, E(H_2, v, u), v)$ in this case.

Case 4: $I/\{u, v\} = \{u := 1, v := 0\}$.

Note that dual(vel) = et and dual(et) = vel (cf. Fig. 2.3). Since $E(x, y, 1) \equiv x \vee y$ and $V(x, y, 0) \equiv x \wedge y$ we conclude that $V(u, E(H_1, v, u), v) \equiv u \wedge (\mathrm{dual}(H_1(x_1, \dots, x_n, y_{j_1}, y_{j_2}, \dots, y_{j_m}, 0, 1)) \vee v) \equiv \mathrm{dual}(H_1(x_1, \dots, x_n, y_{j_1}, y_{j_2}, \dots, y_{j_m}, 0, 1))$ and $V(u, E(H_2, v, u), v) \equiv u \wedge (\mathrm{dual}(H_2(x_1, \dots, x_n, y_{j_1}, y_{j_2}, \dots, y_{j_m}, 0, 1)) \vee v) \equiv \mathrm{dual}(H_2(x_1, \dots, x_n, y_{j_1}, y_{j_2}, \dots, y_{j_m}, 0, 1))$.

Because $(\mathrm{dual}(H_1), \mathrm{dual}(H_2)) \in \mathrm{EQ}^{\mathrm{F}}(\mathrm{dual}(B)) = \mathrm{EQ}^{\mathrm{F}}(B)$ iff $(H_1, H_2) \in \mathrm{EQ}^{\mathrm{F}}(B)$ (note that $\mathrm{dual}(H_1)$ and $\mathrm{dual}(H_2)$ are B-formulas, since $[B]$ is selfdual), we end up with:

$$\begin{aligned} H \in \text{3-TAUT} \ \text{ iff } \ & (H_1(x_1, \dots, x_n, y_{j_1}, \dots, y_{j_m}, 0, 1), \\ & \quad H_2(x_1, \dots, x_n, y_{j_1}, \dots, y_{j_m}, 0, 1)) \in \mathrm{EQ}^{\mathrm{F}}(B) \\ \text{iff } \ & (\mathrm{dual}(H_1(x_1, \dots, x_n, y_{j_1}, \dots, y_{j_m}, 0, 1)), \\ & \quad \mathrm{dual}(H_2(x_1, \dots, x_n, y_{j_1}, \dots, y_{j_m}, 0, 1))) \in \mathrm{EQ}^{\mathrm{F}}(B). \end{aligned}$$

Finally we have by the claims 1-4 that $H \in$ 3-TAUT iff $(V(u, E(H_1, v, u), v), V(u, E(H_2, v, u), v)) \in \mathrm{EQ}^{\mathrm{F}}(B)$ and hence $\mathrm{EQ}^{\mathrm{F}}(B)$ is $\leq_{\mathrm{m}}^{\mathrm{log}}$-hard for **coNP**. By Proposition 4.6.1 we deduce that $\mathrm{EQ}^{\mathrm{C}}(B)$ is $\leq_{\mathrm{m}}^{\mathrm{log}}$-hard for **coNP** as well.

Lower bound for $\mathrm{ISO}^{\mathrm{F}}(B)$. We already know, that if $H \in$ 3-TAUT, then $V(u, E(H_1, v, u), v) \equiv V(u, E(H_2, v, u), v)$. Hence if $H \in$ 3-TAUT, then $(V(u, E(H_1, v, u), v), V(u, E(H_2, v, u), v)) \in \mathrm{ISO}^{\mathrm{F}}(B)$ via $\pi = \mathrm{id}$. First we define

$$\begin{aligned} F_1(x_1, \dots, x_n, y_{j_1}, \dots, y_{j_m}, u, v) =_{\mathrm{def}} V(u, & \\ & E(H_1(x_1, \dots, x_n, y_{j_1}, \dots, y_{j_m}, u, v), v, u), \\ & v), \\ F_2(x_1, \dots, x_n, y_{j_1}, \dots, y_{j_m}, u, v) =_{\mathrm{def}} V(u, & \\ & E(H_2(x_1, \dots, x_n, y_{j_1}, \dots, y_{j_m}, u, v), v, u), \\ & v). \end{aligned}$$

Now conversely assume that $H \notin$ 3-TAUT, but $F_1 \cong F_2$. Hence there is a suitable permutation π such that

$$\begin{aligned} \pi(F_1) &= \pi(V(u, E(H_1(x_1, \dots, x_n, y_{j_1}, \dots, y_{j_m}, u, v), v, u), v)) \\ &= V(\pi(u), \\ &\qquad E(H_1(\pi(x_1), \dots, \pi(x_n), \pi(y_{j_1}), \dots, \pi(y_{j_m}), \pi(u), \pi(v)), \pi(v), \pi(u)), \\ &\qquad \pi(v)) \\ &\equiv V(u, E(H_2(x_1, \dots, x_n, y_{j_1}, \dots, y_{j_m}), v, u), v) \\ &= F_2, \end{aligned}$$

which gives that for all compatible assignments $I \models \pi(F_1)$ iff $I \models F_2$ holds. In the following we prove that no such permutation can exists, giving us the needed contradiction.

Case 1: $\pi(u) \in \{x_1, \ldots, x_n, y_{j_1}, \ldots, y_{j_m}\}$ and $\pi(v) \in \{x_1, \ldots, x_n, y_{j_1}, \ldots, y_{j_m}\}$.
First note that neither $\pi(u) \in \{u, v\}$ nor $\pi(v) \in \{u, v\}$, so we are able to define an assignment I as follows: $I =_{\text{def}} \{\pi(u) := 0, \pi(v) := 0, v := 1, u := 1\} \cup \{x_i := 0 \mid \pi(u) \neq x_i,\ \pi(v) \neq x_i \text{ and } 1 \leq i \leq n\} \cup \{y_{j_l} := 0 \mid \pi(u) \neq y_{j_l},\ \pi(v) \neq y_{j_l} \text{ and } 1 \leq l \leq m\}$. Clearly $I \not\models \pi(F_1)$, but $I \models F_2$. This is a contradiction to the assumption $\pi(F_1) \equiv F_2$.

Case 2: $\pi(u) \in \{x_1, \ldots, x_n, y_{j_1}, \ldots, y_{j_m}\}$ and $\pi(v) \notin \{x_1, \ldots, x_n, y_{j_1}, \ldots, y_{j_m}\}$.
In the next two sub-cases, we consider assignments I such that $I/\{\pi(u), u\} = \{\pi(u) := 0, u := 1\}$ only.

Case 2.1: $\pi(v) = v$.
Now look at all assignments I such that $I/\{v\} = \{v := 1\}$. Clearly all such assignments are models of F_2, since $I/\{u, v\} = \{u := 1, v := 1\}$. Next note that $I \models \pi(F_1)$ iff $I/\{y_{j_1}, \ldots, y_{j_m}, y_{j_1}, \ldots, y_{j_m}\} \models \pi(H_1(x_1, \ldots, x_n, y_{j_1}, \ldots, y_{j_m}, 0, 1))$, because $\pi(v)$ is set to 1 and $\pi(u)$ is set to 0. Clearly $\pi(H_1(x_1, \ldots, x_n, y_{j_1}, \ldots, y_{j_m}, 0, 1))$ is a constant function iff $H_1(x_1, \ldots, x_n, y_{j_1}, \ldots, y_{j_m}, 0, 1)$ is a constant function. But this is a contradiction, since Lemma 4.14 gives us that $H_1(x_1, \ldots, x_n, y_{j_1}, \ldots, y_{j_m}, 0, 1)$ represents not a constant function.

Case 2.2: $\pi(v) = u$.
Analogous to *Case 2.1*.

Case 3: $\pi(v) \in \{x_1, \ldots, x_n, y_{j_1}, \ldots, y_{j_m}\}$ and $\pi(u) \notin \{x_1, \ldots, x_n, y_{j_1}, \ldots, y_{j_m}\}$.
In the next two sub-cases we consider assignments I such that $I/\{\pi(v), v\} = \{\pi(v) := 1, v := 0\}$ only.

Case 3.1: $\pi(u) = u$.
Now look at all assignments I such that $I/\{u\} = \{u := 0\}$. Clearly all such assignments are not models of F_2, since $I/\{u, v\} = \{u := 0, v := 0\}$. Next note that $I \models \pi(F_1)$ iff $I/\{y_{j_1}, \ldots, y_{j_m}, y_{j_1}, \ldots, y_{j_m}\} \models \pi(H_1(x_1, \ldots, x_n, y_{j_1}, \ldots, y_{j_m}, 0, 1))$, because $\pi(v)$ is set to 1 and $\pi(u)$ is set to 0. Clearly $\pi(H_1(x_1, \ldots, x_n, y_{j_1}, \ldots, y_{j_m}, 0, 1))$ is a constant function iff $H_1(x_1, \ldots, x_n, y_{j_1}, \ldots, y_{j_m}, 0, 1)$ is a constant function. But this is a contradiction, since Lemma 4.14 gives us that $H_1(x_1, \ldots, x_n, y_{j_1}, \ldots, y_{j_m}, 0, 1)$ represents not a constant function.

Case 3.2: $\pi(u) = v$.
Analogous to *Case 3.1*.

Case 4: $\pi(v) = u$ and $\pi(u) = v$.
In this case we restrict ourselves to all assignments I such that $I/\{u, v\} = \{u := 1, v := 0\}$. Hence we obtain $I \models \pi(F_1)$ iff $I/\{y_{j_1}, \ldots, y_{j_m}, y_{j_1}, \ldots, y_{j_m}\} \models \pi(H_1(x_1, \ldots, x_n, y_{j_1}, \ldots, y_{j_m}, 0, 1))$ iff $I/\{y_{j_1}, \ldots, y_{j_m}, y_{j_1}, \ldots, y_{j_m}\} \models \text{dual}(H_2(x_1, \ldots, x_n, y_{j_1}, \ldots, y_{j_m}, 0, 1))$ and therefore $|\{I \mid I \models \pi(H_1(x_1, \ldots, x_n, y_{j_1}, \ldots, y_{j_m}, 0, 1))\}| = |\{I \mid I \models \text{dual}(H_2(x_1, \ldots, x_n, y_{j_1}, \ldots, y_{j_m}, 0, 1))\}|$.
Without loss of generality we can assume that the 3-DNF formula H has at least three different negated literals. To achieve this, we can simply replace

3-TAUT by 3-TAUT′ in Lemma 4.14. The problem 3-TAUT′ is defined as follows:

Problem:	3-TAUT′
Instance:	A propositional formula H in 3-DNF, having at least three different negated literals
Question:	Is H a tautology?

Clearly this problem is **coNP**-complete too. Hence H_2 is of the form

$$H_2(x_{j_1},\ldots,x_{j_m},y_{j_1},\ldots,y_{j_m},0,1) \equiv \bigwedge_{l=1}^{m}(x_{j_l} \vee y_{j_l}),\ m \geq 3$$

Now notice that $\#_1(H_2(x_{j_1},\ldots,x_{j_m},y_{j_1},\ldots,y_{j_m},0,1)) = 3^m$ and by Proposition 2.18 $\#_1(\mathrm{dual}(H_2(x_{j_1},\ldots,x_{j_m},y_{j_1},\ldots,y_{j_m},0,1))) = 2^{2m} - 3^m$. By a simple induction we can show that for any $m \geq 3$ the inequality $3^m < 2^{2m} - 3^m$ holds. The formula H_1 is of the form $H' \wedge H_2$ (see Lemma 4.14) and we obtain (cf. Proposition 4.5) that $|\{I \mid I \models \pi(H_1)\}| = |\{I \mid I \models H_1\}| \leq 2^{n-m} \cdot 3^m < 2^{n-m} \cdot (2^{2m} - 3^m) = 2^{n-m} \cdot |\{I \mid I \models \mathrm{dual}(H_2(x_{j_1},\ldots,x_{j_m},y_{j_1},\ldots,y_{j_m},0,1))\}|$ $= |\{I \mid I \models \mathrm{dual}(H_2(x_1,\ldots,x_n,y_{j_1},\ldots,y_{j_m},0,1))\}|$. This shows that $\pi(H_1) \not\equiv H_2$, which is a contradiction to our assumption.

Case 5: $\pi(u) = u$ and $\pi(v) = v$.

Now take only such assignments into account such that $I/\{u,v\} = \{u := 0, v := 1\}$. Hence $I/\{x_1,\ldots,x_n,y_{j_1},\ldots,y_{j_m}\} \models \pi(H_1(x_1,\ldots,x_n,y_{j_1},\ldots,y_{j_m},0,1))$ iff $I \models \pi(F_1)$ iff $I \models F_2$ iff $I/\{x_1,\ldots,x_n,y_{j_1},\ldots,y_{j_m}\} \models H_2(x_1,\ldots,x_n,y_{j_1},\ldots,y_{j_m},0,1)$. But this is a contradiction since Lemma 4.14 gives us that $\pi(H_1(x_1,\ldots,x_n,y_{j_1},\ldots,y_{j_m},0,1)) \not\equiv H_2(x_1,\ldots,x_n,y_{j_1},\ldots,y_{j_m},0,1)$.

In all possible five cases our assumption $F_1 \cong F_2$ results in a contradiction. Hence we conclude $F_1 \not\cong F_2$. The statement for the circuit case follows by Proposition 4.6. □

This leads us to the following classification theorems for the complexity of the equivalence- and isomorphism-problem of B-circuits and B-formulas:

Theorem 4.18. *Let B be a finite set of Boolean functions. The complexity of $\mathrm{EQ}^{\mathrm{C}}(B)$ and $\mathrm{ISO}^{\mathrm{C}}(B)$ is completely characterized by*

if $(B \subseteq \mathrm{N})$ ***then***
  $\mathrm{EQ}^{\mathrm{C}}(B) \in \mathbf{L}$ *and* $\mathrm{ISO}^{\mathrm{C}}(B) \in \mathbf{L}$
else if $(B \subseteq \mathrm{E})$ ***or*** $(B \subseteq \mathrm{V})$ ***then***
  $\mathrm{EQ}^{\mathrm{C}}(B)$ *and* $\mathrm{ISO}^{\mathrm{C}}(B)$ *are* $\leq_{\mathrm{m}}^{\log}$*-complete for* **NL**
else if $(B \subseteq \mathrm{L})$ ***then***
  $\mathrm{EQ}^{\mathrm{C}}(B)$ *and* $\mathrm{ISO}^{\mathrm{C}}(B)$ *are* $\leq_{\mathrm{m}}^{\log}$*-complete for* $\oplus$**L**
else
  $\mathrm{EQ}^{\mathrm{C}}(B)$ *is* $\leq_{\mathrm{m}}^{\log}$*-complete for* **coNP** *and*
  $\mathrm{ISO}^{\mathrm{C}}(B)$ *is* $\leq_{\mathrm{m}}^{\log}$*-hard for* **coNP**.

Proof. The first case $[B] \subseteq \mathrm{N}$ is shown by Lemma 4.8 and the second and third case by Lemma 4.9. Now let B be a finite set of Boolean functions such that $[B] \nsubseteq \mathrm{N}$, $[B] \nsubseteq \mathrm{E}$, $[B] \nsubseteq \mathrm{V}$ and $[B] \nsubseteq \mathrm{L}$. By carefully inspecting Figure 2.2 we conclude that $\mathrm{S}_{00} \subseteq [B]$, $\mathrm{S}_{10} \subseteq [B]$ or $\mathrm{D}_2 \subseteq [B]$ holds in all other cases. By Corollary 4.16 and Theorem 4.17 the statement follows. □

Theorem 4.19. *Let B be a finite set of Boolean functions. The complexity of $\mathrm{ISO}^{\mathrm{F}}(B)$ and of $\mathrm{ISO}^{\mathrm{C}}(B)$ is given by the following dichotomy theorem:*

if *($B \subseteq \mathrm{V}$)* ***or*** *($B \subseteq \mathrm{E}$)* ***or*** *($B \subseteq \mathrm{L}$)* ***then***
$\mathrm{EQ}^{\mathrm{F}}(B) \in \mathbf{L}$ *and* $\mathrm{ISO}^{\mathrm{F}}(B) \in \mathbf{L}$
else
$\mathrm{EQ}^{\mathrm{F}}(B)$ *is $\leq_{\mathrm{m}}^{\log}$-complete for* $\mathbf{coNP}$ *and* $\mathrm{ISO}^{\mathrm{F}}(B)$ *is $\leq_{\mathrm{m}}^{\log}$-hard for* $\mathbf{coNP}$.

Proof. The first part follows by Lemma 4.8 and Theorem 4.12. The second part is shown by using Figure 2.2, Corollary 4.16 and Theorem 4.17. □

Finally it should be mentioned that we used the same reduction to show the **coNP**-hardness of $\mathrm{EQ}^{\mathrm{F}}(B)$ and $\mathrm{ISO}^{\mathrm{F}}(B)$. Therefore it cannot be expected that this reduction is powerful enough to show a better lower bound for the isomorphism-problem. Note that this reduction does not use the ability of permuting variables. Hence it seems possible that a reduction showing a better lower bound for the isomorphism-problem has to take a non-trivial permutation into account.

Another interesting problem arises in the context of the isomorphism-problem of Boolean formulas. It is well-known that the satisfiability-problem for unrestricted Boolean formulas is complete for **NP**, but the satisfiability-problem for monotone Boolean formulas is solvable in **P** (cf. Chapter 3). We showed that in both cases the isomorphism-problem is **coNP**-hard, but it might be possible to show a better upper bound for the isomorphism-problem of monotone formulas ($\mathrm{M}_2 \subseteq [B] \subseteq \mathrm{M}$) than for the isomorphism-problem of unrestricted formulas.

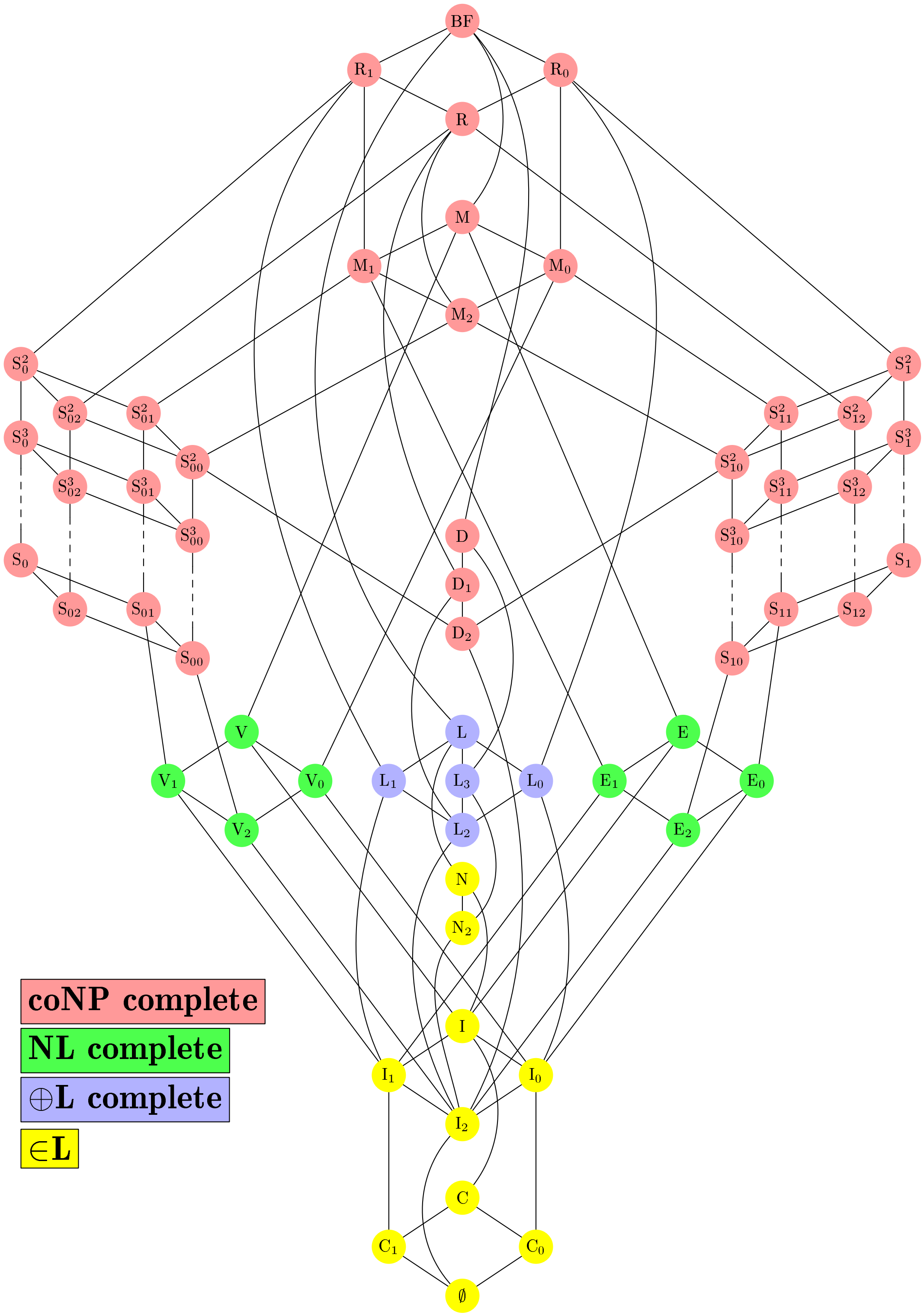


Fig. 4.1. The complexity of $\mathrm{EQ}^{\mathrm{C}}(B)$.

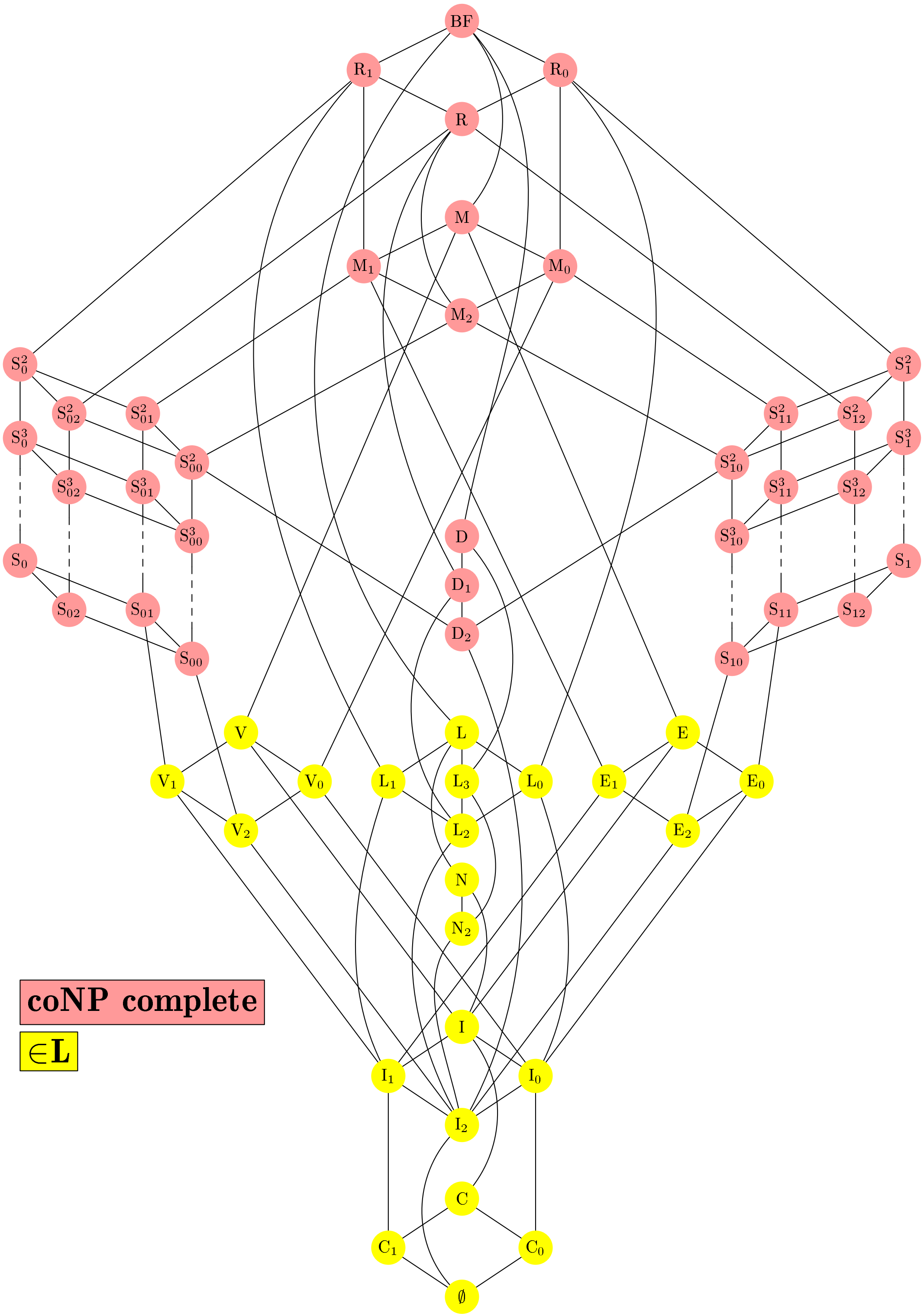


Fig. 4.2. The complexity of $\mathrm{EQ}^{\mathrm{F}}(B)$.

5. Optimization problems

5.1 Introduction

Another interesting and important class of problems is related to the task of finding lexicographically smallest or largest solutions. For example, we are given a propositional formula and are charged with the job of finding the smallest satisfying assignment of this formula. To deal with this kind of problems the class **OptP** was introduced by Krentel in 1988 [Kre88]. While **MaxSNP** and **MinSNP** (cf. [Pap94], pp. 311ff) are defined logically by making use of Fagin's characterization of **NP** [Fag74], the class **OptP** is defined by using Turing machines. It is interesting to know that **OptP** is a superclass of **MaxSNP** and **MinSNP**. The canonical complete problems for **OptP** are the problems LexMax-SAT and LexMinSAT of determining the lexicographically maximal or minimal satisfying assignment of a given (unrestricted) propositional formula. Note that these problems are suspected not to be in **MaxSNP** or **MinSNP**.

We also examine the problem to determine if in the minimal or maximal satisfying assignment the first variable is set to true. These problems are known as OddLexMinSAT and OddLexMaxSAT, and were shown to be $\mathbf{P}^{\mathbf{NP}}$-complete for unrestricted propositional formulas by Wagner [Wag87]. For an exact definition we refer to Sect. 5.3.2 and Sect. 5.4.4. Considering formulas given by a set of Boolean connectives or constraints, we obtain dichotomy theorems for different variations of OddLexMinSAT and OddLexMaxSAT, showing that these problems are either complete for $\mathbf{P}^{\mathbf{NP}}$ or polynomial-time solvable.

In contrast to the previous chapters, we will study the problem of finding minimal and maximal satisfying assignments of two different kinds of generalized propositional formulas here. In Sect. 5.3 we will focus on B-formulas, whereas we will cope with so called S-CSPs in Sect. 5.4. S-CSPs were introduced by Thomas Schaefer in 1978 (cf. [Sch78]) and regained interest in the last years. For a short overview we refer to Sect. 5.4.1. We can regard S-CSPs as generalized propositional formulas in conjunctive normalform, where each clause is a symbol for a Boolean function out of S. We will show for example that if constants are allowed in S-CSPs, then the problem of deciding satisfiability is **NP**-complete if and only if the problem of finding the smallest assignment is **OptP**-complete. (In the case that constants are forbidden, an analogous result does not hold unless $\mathbf{P} = \mathbf{NP}$.)

In contrast to this, if constants are forbidden then we completely identify those cases where the optimization problem is hard and the decision problem is easy.

After presenting some notations and preliminary results in Sect. 5.2.1 and Sect. 5.2.2, we will turn to B-formulas in the Post-context in Sect. 5.3. In Sect. 5.4 we will advance with Schaefer's constraint satisfaction problems.

Note that the results of this chapter are based on [RV00].

5.2 Maximization and minimization problems

5.2.1 Ordered assignments

As mentioned in the introduction we want to talk about lexicographically smallest (biggest, resp.) satisfying assignments and we want to compare truth assignments lexicographically. Because of that we have to talk about the first, second, etc. variable. In the following we will additionally assume an ordering on V without further mention.

An *ordering on the variables* induces an *ordering on assignments* as follows: $(a_1, \ldots, a_k) < (b_1, \ldots, b_k)$ if and only if there is an $i \leq k$ such that for all $j < i$ we have $a_j = b_j$ and $a_i < b_i$. We refer to this ordering as the *lexicographical ordering*. We write $(a_1, \ldots, a_k) \models_{\min} \Phi$ ($(a_1, \ldots, a_k) \models_{\max} \Phi$, resp.) iff $(a_1, \ldots, a_k) \models \Phi$ and there exists no lexicographically smaller (larger, resp.) $(a'_1, \ldots, a'_k) \in \{0,1\}^k$ such that $(a'_1, \ldots, a'_k) \models \Phi$.

5.2.2 The class OptP

The study of optimization problems in computational complexity theory started with the work of Krentel [Kre88, Kre92]. He defined the class **OptP** and an oracle hierarchy built on this class using so called *metric Turing machines*. We do not need this machine model here; therefore we proceed by defining the class **OptP** and its subclasses **MinP** and **MaxP** (explicitly introduced the first time in [Köb89]) using a characterization given in [VW95].

We fix the alphabet $\Sigma = \{0, 1\}$. Let **FP** denote the class of all functions $f\colon \Sigma^* \to \Sigma^*$ computable deterministically in polynomial time. Using one of the well-known bijections between Σ^* and the set of natural numbers (e.g., dyadic encoding) we may also think of **FP** (and the other classes of functions defined below) as a class of *number-theoretic functions.*

Definition 5.1. A function h belongs to the class **MinP** if there is a function $f \in$ **FP** and a polynomial p such that, for all x,

$$h(x) = \min_{|y|=p(|x|)} f(x, y),$$

where minimization is taken with respect to lexicographical order. The class **MaxP** is defined by taking the maximum of these values. Finally, let **OptP** = **MinP** $\cup$ **MaxP**.

OptP is a subclass of $\mathbf{FP}^{\mathbf{NP}}$, and it is well known that the closure of all three classes **MinP**, **MaxP**, and **OptP** under 1-Turing reductions coincides with the class $\mathbf{FP}^{\mathbf{NP}}$; which means that showing completeness of a problem for **MinP** generally implies hardness of the same problem for **MaxP** and completeness for **OptP**, see [Kre88, VW95, Vol94].

Krentel in [Kre88] presented a number of problems complete for **OptP** under 1-Turing reducibility. The most important one for us is the problem of finding the lexicographically minimal satisfying assignment of a given formula, defined as follows:

PROBLEM:	LexMin3-SAT
INSTANCE:	A propositional formula Φ in 3-CNF
OUTPUT:	The lexicographically smallest satisfying assignment of Φ, or "$\perp$" if Φ is unsatisfiable

The problem LexMax3-SAT is defined analogously.

Proposition 5.2 ([Kre88]). *The problems* LexMin3-SAT *and* LexMax3-SAT *are* $\leq_{1\text{-T}}^{log}$*-complete for* **OptP**.

5.3 Finding optimal assignments of B-formulas

5.3.1 Dichotomy theorems for LexMinSAT(B) and LexMaxSAT(B)

In this section we study the complexity of finding the lexicographical smallest (largest, resp.) satisfying assignment of B-formulas, for all finite sets B of Boolean functions; formally:

PROBLEM:	LexMinSAT(B)
INSTANCE:	A B-formula Φ
OUTPUT:	The lexicographically smallest satisfying assignment of Φ, or "$\perp$" if Φ is unsatisfiable

The corresponding maximization problem is denoted by LexMaxSAT(B).

The cases of formulas with easy LexMin/MaxSAT-problem are easy to identify:

Lemma 5.3. *If B is a finite set of Boolean functions such that $B \subseteq \mathrm{R}_1$, then* LexMaxSAT$(B) \in$ **FP**.

Proof. If $\Phi(x_1, \ldots, x_n)$ is a B-formula, then $f_\Phi(1^n) = 1$. □

Lemma 5.4. *If B is a finite set of Boolean functions such that $B \subseteq \mathrm{M}$ or $B \subseteq \mathrm{L}$, then* LexMinSAT$(B) \in$ **FP** *and* LexMaxSAT$(B) \in$ **FP**.

Proof. The following is a slight modification of an algorithm presented in [CH97] (cf. Fig.5.1):

In both cases $\mathrm{SAT}^{\mathrm{F}}(B) \in \mathbf{P}$, because $\mathrm{SAT}^{\mathrm{C}}(B) \in \mathbf{P}$ and $\mathrm{SAT}^{\mathrm{F}}(B) \leq_{\mathrm{m}}^{\log} \mathrm{SAT}^{\mathrm{C}}(B)$ (cf. Theorem 3.8 and Theorem 2.3). Hence, given Φ, a minimal satisfying assignment can be computed as follows: Consider all variables x in Φ in their order, first set x to false and if the resulting formula is satisfiable then proceed with $x :=$ false to the next variable. If the resulting formula is not satisfiable, then we set x to true and test satisfiability. If now the result is satisfiable, we proceed (with $x :=$ true) to the next variable, otherwise we output $\perp$. $\square$

```
function calc_smallest_satisfying_assignment(Φ(x_1, ..., x_n));
begin
      e := Φ;
      if (Φ is satisfiable) then begin
         for  i := 1 to n do begin
               e' := e;
               replace x_i in e' by constant 0;
               if (e' is satisfiable) then begin
                  A[i] := 0;
                  e := e'
               end
               else begin
                  A[i] := 1;
                  replace x_i in e by constant 1
               end
         end;
         writeln(A)
      end
      else
         writeln(⊥)
end.
```

Fig. 5.1. An algorithm to calculate the lexicographically minimal satisfying assignment.

We will now show that in all other cases, computing minimal or maximal assignments is hard, i.e., complete for **OptP**. Clearly, for all B, $\mathrm{LexMinSAT}(B) \in$ **MinP** ($\mathrm{LexMaxSAT}(B) \in$ **MaxP**, resp.). Hence, in the subsequent theorems we have to prove hardness via suitable reductions.

The following easy lemma shows that we can construct, for any 3-CNF-formula, a logically equivalent $(B \cup \{0,1\})$-formula, if B together with the constants 0 and 1 forms a complete base. This is very similar to Lemma 4.13.

Lemma 5.5. *Let $k > 0$ be fixed and B be a finite set of Boolean functions such that there are B-formulas $E(x,y,v,u)$ and $N(x,v,u)$ with $E(x,y,0,1) \equiv x \wedge y$ and $N(x,0,1) \equiv \overline{x}$. Then, for any k-CNF formula $\Phi(x_1,\ldots,x_n)$, there exists a B-formula $\Psi(x_1,\ldots,x_n,u,v)$ such that $\Psi(x_1,\ldots,x_n,0,1) \equiv \Phi(x_1,\ldots,x_n)$. Moreover, Ψ can be computed from Φ in logarithmic space.*

Proof. Since $\{\wedge,\neg\}$ forms a complete basis, we can trivially transform every k-CNF formula into a $(B\cup\{0,1\})$-formula. In the case of a given CNF-formula with m clauses, to avoid an exponential blowup while replacing the m $\wedge$'s, we insert parentheses in such a way that we get a tree of depth $\log m$, and then replace each $\wedge$ by an appropriate $(B \cup \{0,1\})$-formula. Finally, the $k-1$ $\vee$'s inside the clauses are replaced by corresponding B-formulas. $\square$

For some bases B, the constants 0 and 1 needed in the preceding lemma are easy to construct. This is the contents of the following theorem.

Theorem 5.6. *If B is a finite set of Boolean functions such that $[B] \supseteq \mathrm{S}_1$, then* LexMinSAT$(B)$ *(*LexMaxSAT(B)*, resp.) is $\leq^{log}_{1\text{-T}}$-complete for* **MinP** *(***MaxP***, resp.).*

Proof. From Theorem 2.11 we know that the function $g(x,y) = x \wedge \overline{y}$ is a base for S_1. It is obvious that $g(x, g(x,y)) = x \wedge y$ and $g(1,x) = \overline{x}$. Since $[B] \supseteq \mathrm{S}_1$ there exist B-formulas $G(x,y) \equiv g(x,y)$, $E(x,y,v,u) \equiv x \wedge y$ and $N(x,u,1) \equiv \overline{x}$. Note that the variables u and v are not used in E and that u is not used in N.

Let $\Phi(x_1,\ldots,x_n)$ be a 3-CNF formula. We use Lemma 5.5 to obtain $\Psi'(x_1,\ldots,x_n,u,v)$ such that $\Phi(x_1,\ldots,x_n) \equiv \Psi'(x_1,\ldots,x_n,u,1)$. Now let $\Psi(x_1,\ldots,x_n,u,v) =_{\text{def}} E(\Psi'(x_1,\ldots,x_n,u,v),v)$. Clearly $I \models_{\min} \Phi$ iff $I \cup \{u := 0, v := 1\} \models_{\min} \Psi$ ($I \models_{\max} \Phi$ iff $I \cup \{u := 1, v := 1\} \models_{\max} \Psi$, resp.).

Let $g_2 \in \mathbf{FL}$ compute the above transformation of Φ into Ψ. From an assignment for Ψ, remove the variables u and v by function $g_1 \in \mathbf{FL}$; or simply output $\perp$ if Ψ is unsatisfiable. The functions g_1, g_2 witness LexMin3-SAT $\leq^{log}_{1\text{-T}}$ LexMinSAT(B) (LexMax3-SAT $\leq^{log}_{1\text{-T}}$ LexMaxSAT(B)). $\square$

For other bases, the constants are not obtained so easily. Next we show how to exploit the order of variables in such a way that the minimal model of a formula is forced to assign the values 0 and 1 to particular variables. We will then use these variables as a replacement for the constants needed in Lemma 5.5.

Lemma 5.7. *Let B be a finite set of Boolean functions. If there are B-formulas $E(x,y,v,u)$, $N(x,v,u)$ and $F(v,u,x)$ such that $E(x,y,0,1) \equiv x \wedge y$, $N(x,0,1) \equiv \overline{x}$, $F(0,0,x) \equiv 0$, $F(0,1,x) \equiv x$ and not $F(1,0,x) \equiv F(1,1,x) \equiv 0$, then* LexMinSAT$(B)$ *is $\leq^{log}_{1\text{-T}}$-complete for* **MinP**.

Proof. We show that LexMin3-SAT $\leq^{log}_{1\text{-T}}$ LexMinSAT(B). Let $\Phi(x_1,\ldots,x_n)$ be a 3-CNF formula. The function g_2 for the 1-Turing reduction works as described below:

Since there exist B-formulas E and N we can use Lemma 5.5 to obtain $\Psi'(x_1,\ldots,x_n,v,u)$ in logarithmic space such that $\Phi(x_1,\ldots,x_n) \equiv \Psi'(x_1,\ldots,x_n,0,1)$. Let be $\Psi(x_1,\ldots,x_n,v,u) =_{\text{def}} F(v,u,\Psi')$. The variables are ordered by $v < u < x_1 < x_2 < \cdots < x_n$.

Claim 1: $I \models_{\min} \Phi$ iff $I \cup \{v := 0, u := 1\} \models_{\min} \Psi$.

"$\Rightarrow$": Let $I \models_{\min} \Phi$. Any $I' \models \Psi$ does not assign $u = v = 0$, so if there exists an $I' < I \cup \{v := 0, u := 1\}$ then $I'/\{u,v\} = \{v := 0, u := 1\}$ and $I'/\{x_1,\ldots,x_n\} < I$. But this is a contradiction since $I'/\{x_1,\ldots,x_n\} \models \Phi$ (recall that $\Psi(x_1,\ldots,x_n,0,1) \equiv \Phi(x_1,\ldots,x_n)$).

"$\Leftarrow$": Let $I \cup \{v := 0, u := 1\} \models_{\min} \Psi$. If there is an assignment $I' < I$ and $I' \models \Phi$ then $I' \cup \{v := 0, u := 1\} \models \Psi$, which is a contradiction since $I' \cup \{v := 0, u := 1\} < I \cup \{v := 0, u := 1\}$.

Claim 2: Φ is unsatisfiable iff $I' \models_{\min} \Psi$ where $I' \geq \{x_1 = x_2 = \cdots = u := 0, v := 1\}$.

First note that Ψ is satisfiable since $F(1,0,\Xi) \not\equiv 0$ or $F(1,1,\Xi) \not\equiv 0$. So there exists an assignment $I \models \Psi$.

"$\Rightarrow$": Let $\Phi(x_1, \ldots, x_n)$ be unsatisfiable. Suppose that there is an $I' \models \Psi$ such that $I' < \{x_1 = x_2 = \cdots = x_n = u := 0, v := 1\}$ then $I'/\{x_1, x_2, \ldots, x_n\} \models \Phi$, because I' cannot assign $v = 0$ and $u = 0$. But this is a contradiction since Φ is not satisfiable.

"$\Leftarrow$": Let $I' \models_{\min} \Psi$ and $I' \geq \{x_1 = x_2 = \cdots = u := 0, v := 1\}$. If $I \models \Phi$ then $I \cup \{v := 0, u := 1\} < \{x_1 = x_2 = \cdots = x_n = u := 0, v := 1\}$ and $I \cup \{v := 0, u := 1\} \models \Psi$ which is a contradiction since $I' \models_{\min} \Psi$.

Now the function g_1 of the 1-Turing reduction simply outputs "$\perp$" if it gets an assignment I' greater than or equal to $\{x_1 = x_2 = \cdots = u := 0, v := 1\}$ as an input. In all other cases it removes the assignments for v and u and outputs the resulting assignment. This proves LexMin3-SAT $\leq^{log}_{1\text{-T}}$ LexMinSAT(B). □

For various non complete sets B one can show that formulas E, N, and F, as required in the above lemma, exist. Hence we obtain complexity results for more non complete classes of Boolean functions. As we pointed out before the statement of the lemma, in this case we do depend on the order of the variables (in contrast to Theorem 5.6).

Theorem 5.8. *If B is a finite set of functions such that $[B] \supseteq \mathrm{S}_{02}$, $[B] \supseteq \mathrm{S}_{12}$ or $[B] \supseteq \mathrm{D}_1$, then* LexMinSAT$(B)$ *is $\leq^{log}_{1\text{-T}}$-complete for* **MinP**.

Proof.

Case 1: $[B] \supseteq \mathrm{S}_{02}$.

From Theorem 2.11 we know that $g(x,y,z) = x \vee (y \wedge \overline{z})$ is a base for S_{02}. Let be $g'(x,y) =_{\text{def}} g(0,x,y) = x \wedge \overline{y}$. Clearly $g'(x, g'(x,y)) = x \wedge y$ and $g'(1,x) = \overline{x}$. Since $[B] \supseteq \mathrm{S}_{02}$ there exist B-formulas $E(x,y,v,u)$ and $N(x,v,u)$ such that $E(x,y,0,u) \equiv x \wedge y$ (again the variable u is not used in E) and $N(x,0,1) \equiv \overline{x}$. Moreover note that $g(x,y,g(x,y,z)) = x \vee (y \wedge z)$ and therefore there exists a B-formula F such that $F(v,u,z) \equiv v \vee (u \wedge z)$. We obtain $F(0,0,x) \equiv 0$, $F(0,1,x) \equiv x$, $F(1,0,x) \equiv 1$ and $F(1,1,x) \equiv 1$. Using Lemmas 5.5 and 5.7 the statement follows.

Case 2: $[B] \supseteq \mathrm{S}_{12}$.

From Theorem 2.11 we know that $g(x,y,z) = x \wedge (y \vee \overline{z})$ is a base for S_{12}. Obviously $g(x,y,1) = x \wedge y$, $g(1,0,x) = \overline{x}$ and $g(x,y,g(x,y,z)) = x \wedge (y \vee z)$. Hence there exist B-formulas $E(x,y,v,u)$, $N(x,v,u)$ and $F(v,u,z)$ such that $E(x,y,0,1) \equiv x \wedge y$, $N(x,0,1) \equiv \overline{x}$ and $F(v,u,z) \equiv u \wedge (v \vee z)$. Hence, $F(0,0,x) \equiv 0$, $F(0,1,x) \equiv x$, $F(1,0,x) \equiv 0$ and $F(1,1,x) \equiv 1$. Using Lemmas 5.5 and 5.7 the statement follows.

Case 3: $[B] \supseteq \mathrm{D}_1$.

From Theorem 2.11 we know that $g(x, y, z) = (x \wedge y) \vee (x \wedge \overline{z}) \vee (y \wedge \overline{z})$ is a base for D_1. We obtain $g(x, y, 1) = x \wedge y$, $g(0, 1, x) = \overline{x}$ and $g(x, y, g(x, y, z)) = (x \wedge y) \vee (x \wedge z) \vee (y \wedge z)$. Since $[B] \supseteq \mathrm{D}_1$ we know that there exists B-formulas $E(x, y, v, u)$, $N(x, v, u)$ and $F(v, u, x)$ such that $E(x, y, 0, 1) \equiv x \wedge y$, $N(x, 0, 1) \equiv \overline{x}$ and $F(v, u, z) \equiv (v \wedge u) \vee (v \wedge z) \vee (u \wedge z)$. Hence, $F(0, 0, x) \equiv 0$, $F(0, 1, x) \equiv x$ and $F(1, 1, x) \equiv 1$. Using Lemmas 5.5 and 5.7 the statement follows. □

Combining Lemma 5.4 and Theorem 5.8, we are now ready to prove a dichotomy theorem for LexMinSAT(B) for arbitrary finite sets of Boolean functions B:

Corollary 5.9 (Dichotomy Theorem for LexMinSAT(B)**).** *Let B be a finite set of Boolean functions. If $[B] \supseteq \mathrm{S}_{02}$, $[B] \supseteq \mathrm{S}_{12}$ or $[B] \supseteq \mathrm{D}_1$, then* LexMin-SAT($B$) *is $\leq^{log}_{\text{1-T}}$-complete for* **MinP**. *In all other cases* LexMinSAT(B) $\in$ **FP**.

Proof. The first part of the statement is proved by Theorem 5.8. Now let B be a finite set of Boolean functions such that $[B] \not\supseteq \mathrm{S}_{02}$, $[B] \not\supseteq \mathrm{S}_{12}$ and $[B] \not\supseteq \mathrm{D}_1$. By inspecting Figure 2.2 we obtain that either $B \subseteq \mathrm{M}$ or $B \subseteq \mathrm{L}$. Using Lemma 5.4 the second part of the statement follows. □

Turning now to LexMaxSAT(B), we observe that, if non $\in [B]$, the problem of finding the minimal satisfying assignment of a B-formula reduces to the problem of determining the maximal satisfying assignment of a B-formula. We will use this to show that LexMaxSAT(B) is hard for **OptP** if B is the set of selfdual functions (D).

Lemma 5.10. *Let B be a finite set of Boolean functions. If* non $\in [B]$, *then* LexMinSAT(B) $\leq^{log}_{\text{1-T}}$ LexMaxSAT(B).

Proof. Let $\Phi(x_1, \ldots, x_n)$ be an arbitrary B-formula. We define the B-formula $\Phi^{\mathrm{R}}(x_1, \ldots, x_n) =_{\text{def}} \Phi(\overline{x}_1, \ldots, \overline{x}_n)$. Note that Φ^{R} is a B-formula since non $\in [B]$. Clearly $I \models_{\min} \Phi$ iff $\overline{I} \models_{\max} \Phi^{\mathrm{R}}$.

The 1-Turing reduction now works as follows: First g_2 replaces any occurrence of x_i by a formula representing non(x), for $1 \leq i \leq n$, resulting in Φ^{R}. The function g_1 simply maps $\perp$ to $\perp$ and any assignment I to $\overline{I}$. □

Now we are able to prove a Dichotomy Theorem for LexMaxSAT(B).

Corollary 5.11 (Dichotomy Theorem for LexMaxSAT(B)**).** *Let B be a finite set of Boolean functions. If $[B] \supseteq \mathrm{S}_1$ or $[B] \supseteq \mathrm{D}$, then* LexMaxSAT(B) *is $\leq^{log}_{\text{1-T}}$-complete for* **MaxP**. *In all other cases* LexMaxSAT(B) $\in$ **FP**.

Proof. The case $[B] \supseteq \mathrm{S}_1$ is given in Theorem 5.6.

Now let $[B] \supseteq \mathrm{D}$. From Theorem 2.11 we know that $g(x, y, z) = (x \wedge \overline{y}) \vee (x \wedge \overline{z}) \vee (\overline{y} \wedge \overline{z})$ is a base of D. Clearly there exists a B-formula $N(x) \equiv \overline{x}$, because $g(y, y, x) = \overline{x}$. Since LexMaxSAT(D) $\in$ **MaxP**, LexMinSAT(D) is $\leq^{log}_{\text{1-T}}$-complete

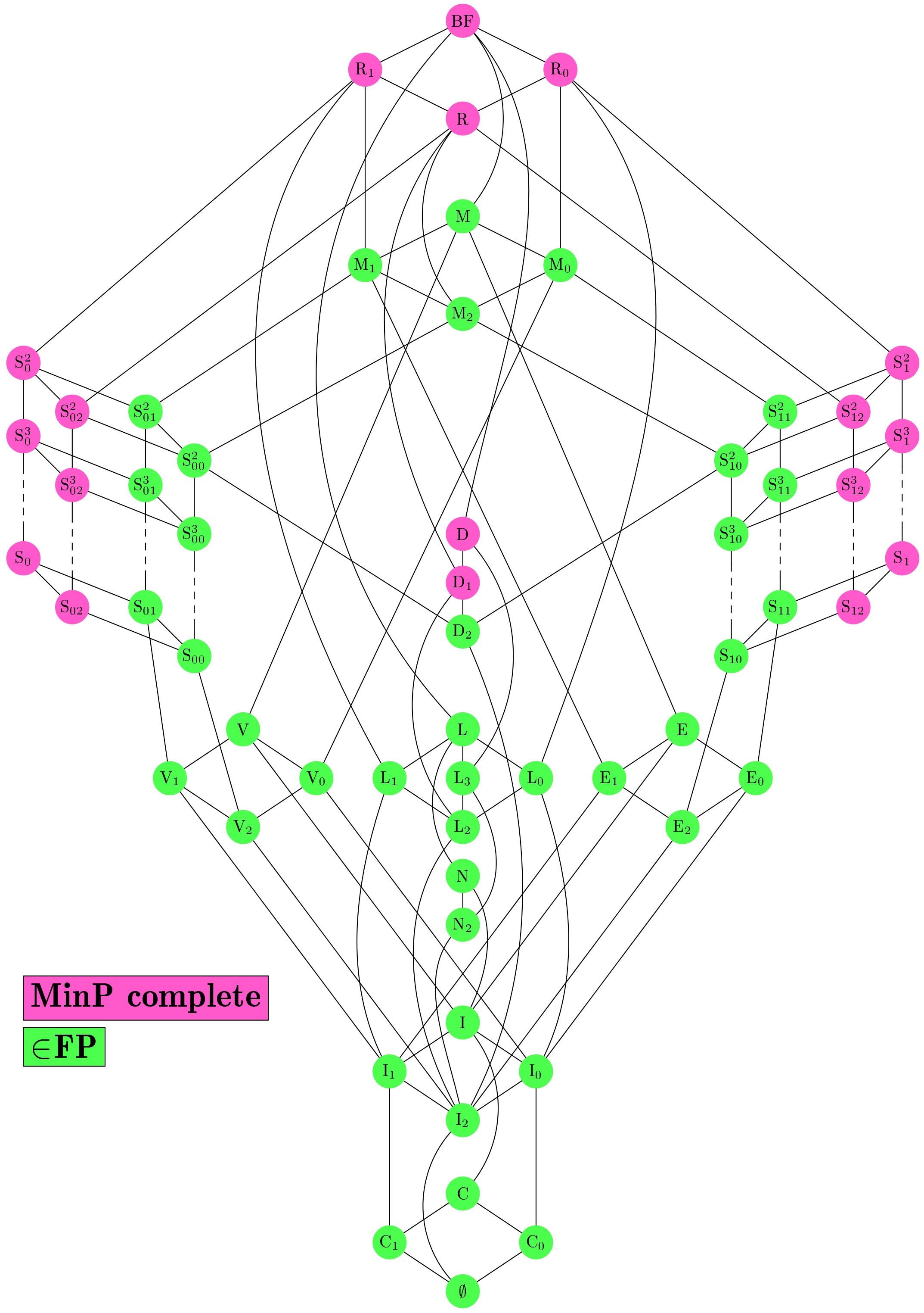
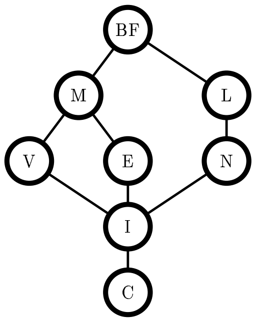


Fig. 5.2. The complexity of LexMinSAT(B).

for **MinP** (see Corollary 5.9) and LexMinSAT(D) $\leq^{log}_{1\text{-T}}$ LexMaxSAT(D), we conclude that LexMaxSAT(D) is $\leq^{log}_{1\text{-T}}$-complete for **MaxP**.

Finally, let B be a finite set of Boolean functions such that $[B] \not\supseteq \mathrm{S}_1$ and $[B] \not\supseteq$ D. Using Figure 2.2 we see that B must be monotone, linear or 1-reproducing. From Lemma 5.4 and Proposition 5.3 it follows that LexMaxSAT(B) $\in$ **FP**. $\square$



Finally, let us remark that a dichotomy result for B-formulas with constants, i.e., $0, 1 \in [B]$, is much easier. Recall that the graph of all closed classes containing the constants (but not necessarily the identity functions) is given to the right (cf. Section 3.2). The classes V, E and M contain all vel, et and monotone functions, whereas N and L contain the non functions and the linear functions. The class I consists of all identity functions, the class C of all constant functions. This graph can easily be found by adding the constant 0-ary functions 0 and 1 to all classes shown in Figure 2.2 and using Posts techniques to identify the resulting classes (cf. Examples 2.8 and 2.9).

Now by the techniques of Lemma 5.4 and Lemma 5.5 the following dichotomy result for LexMinSAT($B \cup \{0,1\}$) and LexMaxSAT($B \cup \{0,1\}$) can be deduced:

Corollary 5.12. *Let B be a finite set of Boolean functions, such that $[B \cup \{0,1\}]$ = BF. Then* LexMinSAT($B \cup \{0,1\}$) *(*LexMaxSAT($B \cup \{0,1\}$)*, resp.) is* $\leq^{log}_{1\text{-T}}$-*complete for* **MinP** *(***MaxP***, resp.). In all other cases* LexMinSAT($B \cup \{0,1\}$) $\in$ **FP** *(*LexMaxSAT($B \cup \{0,1\}$) $\in$ **FP***, resp.).*

5.3.2 Completeness results for the class $\mathbf{P}^{\mathbf{NP}}$

Given a function $f\colon \Sigma^* \to \Sigma^*$, define the set $L_f = \{x \in \Sigma^* \mid$ the last bit of $f(x)$ is a $1\}$. Often it turns out that if f is complete for **OptP** under 1-Turing reductions, then the set L_f is complete for $\mathbf{P}^{\mathbf{NP}}$ under usual many-one reductions; a precise statement is given below.

In our context the above problem translates to the question if the lexicographically minimal (maximal, resp.) satisfying assignment of a given B-formula is odd. We will call this problem OddLexMinSAT(B) (OddLexMaxSAT(B), resp.). More precisely:

PROBLEM: OddLexMinSAT(B)
INSTANCE: A B-formula $\Phi(x_1, \ldots, x_n)$
QUESTION: Is in the lexicographically smallest satisfying assignment of Φ the variable x_n set to 1?

The corresponding problems for unrestricted propositional formulas will be denoted by OddLexMinSAT and OddLexMaxSAT. The following result is known:

Proposition 5.13 ([Wag87]). OddLexMinSAT *and* OddLexMaxSAT *are complete for the class* $\mathbf{P}^{\mathbf{NP}}$ *under many-one reductions.*

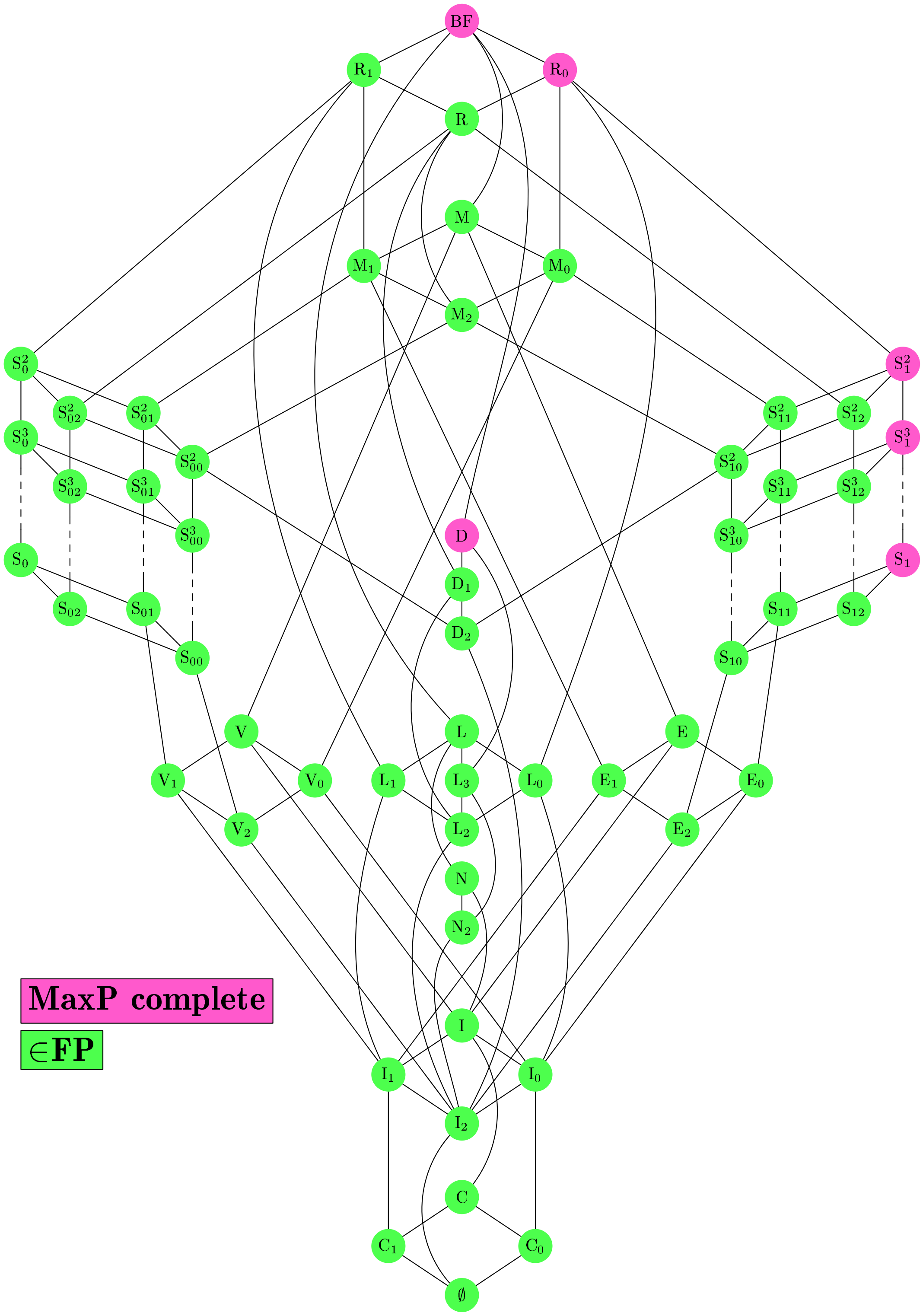


Fig. 5.3. The complexity of LexMaxSAT(B).

If f is complete for **MinP** or **MaxP** under functional many-one reductions (that is, 1-Turing reductions where the outer function g_1 is the identity), then L_f is complete for $\mathbf{P}^{\mathbf{NP}}$ under usual many-one reductions [Kre88], see also [Vol94]. In the case that f is only 1-Turing complete a similar result is not known. However, by a separate proof, the complexity of OddLexMinSAT(B) and OddLexMax-SAT(B) can be determined.

Next we give a Dichotomy Theorem for OddLexMinSAT(B) and OddLexMax-SAT(B).

Theorem 5.14. *For any finite set B of Boolean functions,* OddLexMinSAT(B) *(*OddLexMaxSAT(B)*, resp.) is complete for* $\mathbf{P}^{\mathbf{NP}}$ *iff* LexMinSAT(B) *(*LexMax-SAT(B)*, resp.) is complete for* **MinP** *(***MaxP***, resp.).*

Proof. Observe that in the case that LexMinSAT(B) $\in$ **FP** (LexMaxSAT(B) $\in$ **FP**, resp.) we are able to compute the lexicographically smallest (largest, resp.) solution in polynomial time. Because of this we can clearly decide OddLexMin-SAT(B) (OddLexMaxSAT(B), resp.) in polynomial time by checking the value of the most significant bit in this assignment.

Clearly OddLexMinSAT(B) $\in \mathbf{P}^{\mathbf{NP}}$ (OddLexMaxSAT(B) $\in \mathbf{P}^{\mathbf{NP}}$, resp.). Now let LexMinSAT(B) (LexMaxSAT(B), resp.) be complete for **MinP** (**MaxP**, resp.). In these cases we show how to reduce OddLexMin3-SAT to OddLexMin-SAT(B) (OddLexMax3-SAT to OddLexMaxSAT(B), resp.). Note that in all these cases $[B \cup \{0,1\}] = \mathrm{BF}$. Hence there exists a B-formula $A(x, y, v, u)$ such that $A(x, y, 0, 1) \equiv x \leftrightarrow y$. Now modify the reduction function g_2 of Lemma 5.7 such that we obtain a formula $\Psi = F(u, v, (\Psi' \wedge A(x_n, z, v, u)))$, where z is larger than any other variable used in this construction. Now for any 3-CNF-formula Φ where $I \models_{\min} \Phi$ and $I' \models_{\min} \Psi$ it holds that $I(x_n) = I'(x_n) = I(z)$. So $\Phi \in$ OddLexMin3-SAT iff $\Psi \in$ OddLexMinSAT(B). By the same modification of the reduction function in Theorem 5.6 we can show that $\Phi \in$ OddLexMax-3-SAT iff $\Phi' \in$ OddLexMaxSAT(B) holds. $\square$

5.4 Constraint satisfaction problems

5.4.1 Introduction

In 1978 Thomas J. Schaefer proved a remarkable result. He examined satisfiability of propositional formulas for certain syntactically restricted formula classes. Each such class is given by a set S of Boolean functions allowed when constructing formulas. An *S-formula* in his sense is a conjunction of clauses, where each clause consists of a Boolean function from S applied to some propositional variables. Such a Boolean function can be interpreted as a constraint that has to be fulfilled by a given assignment; the satisfiability problem for S-formulas hence provides a mathematical model for the examination of the complexity of constraint satisfaction problems. Let CSP(S) denote the problem to decide for a given S-formula if it is satisfiable. Schaefer showed that, depending on S, the problem CSP(S) is either (1) efficiently solvable (i. e., in polynomial time) or (2) **NP**-complete; and he gave a simple criterion that, given S, allows one to determine whether (1) or (2) holds. Since the complexity of CSP(S) is either easy or hard (and not located in one of the – under the assumption $\mathbf{P} \neq \mathbf{NP}$ – infinitely many intermediate degrees between **P** and the **NP**-complete sets [Lad75]), Schaefer called this a "*dichotomy theorem for satisfiability*".

In the last few years, these results regained interest among complexity theorists. Constraint satisfaction problems were studied by Nadia Creignou and others [Cre95, CH96, CH97], see also the monograph [CKS99]. Considering different versions of satisfiability, optimization and counting problems, dichotomy theorems for classes as **NP**, **MaxSNP**, and **#P** were obtained. Also, the study of Schaefer's formulas lead to remarkable results about approximability of optimization problems in the constraint satisfaction context [KST97, KSW97].

In this section, we continue this line of research by considering the complexity of determining the lexicographically minimal (maximal, resp.) satisfying assignment of a given S-formula. In the case of unrestricted formulas, these problems are known to be complete for Krentel's class **OptP** [Kre88] (cf. Section 5.2.2). The main results, presented in this section, is a clarification of the complexity of the problem to determine maximal or minimal satisfying assignments of formulas given by a set of constraints. We will show that in all cases, the considered problem is either complete for **OptP** or solvable in polynomial time, depending on the set of allowed connectives/constraints.

The problem of maximizing (or minimizing) the number of clauses satisfied in (unrestricted) propositional formula is complete for the class **MaxSNP** (or **MinSNP**). In 1995 Nadia Creignou examined this problem for S-formulas. Interestingly she also obtained a dichotomy theorem: She proved that this problem is either polynomial-time solvable or **MaxSNP**-complete, depending on properties of S [Cre95] (In 1997 the approximability of this problem and the corresponding minimization problem was examined in [KSW97, KST97], leading to a number of deep results.). The complexity of counting problems and enumeration problems based on satisfiability of S-formulas was studied in [CH96, CH97].

5.4.2 Preliminaries

Let S be a set of Boolean functions. In this section we will always assume that such S are nonempty and finite. *S-formulas in the Schaefer sense*, or, *S-CSPs*, will now be propositional formulas consisting of clauses built by using functions from S applied to arbitrary variables. Formally, let $S = \{f_1, f_2, \dots, f_n\}$ be a set of Boolean functions and V be a set of variables. An *S-CSP* Φ (over V) is a finite conjunction of clauses $\Phi = C_1 \wedge \dots \wedge C_k$, where each C_i is of the form $\tilde{f}(x_1, \dots, x_k)$, $f \in S$, $\tilde{f}$ is the symbol representing f, k is the arity of f, and $x_1, \dots, x_k \in V$. If some variables of an S-CSP Φ are replaced by the constants 0 or 1 then this new formula Φ' is called *S-CSP with constants*. If $\Phi = C_1 \wedge \dots \wedge C_k$ is a CSP over

V, and I is an assignment with respect to V, then $I \models \Phi$ if Φ satisfies all clauses C_i. Here, a clause $\tilde{f}(x_1, \ldots, x_k)$ is satisfied, if $f\big(I(x_1), \ldots, I(x_k)\big) = 1$. By $\Phi\left[\begin{smallmatrix} x \\ y \end{smallmatrix}\right]$ we denote the formula created by simultaneously replacing each occurrence of x in Φ by y, where x, y are either variables or constants.

We will consider different types of Boolean functions, following the terminology of Schaefer [Sch78].

- The Boolean function f is called *0-valid* if $f(0, \ldots, 0) = 1$ and *1-valid* if $f(1, \ldots, 1) = 1$.
- The Boolean function f is called *Horn* (*anti-Horn*, resp.) if f is represented by a CNF formula having at most one unnegated (negated, resp.) variable in any conjunct.
- A Boolean function f is called *bijunctive* if it is represented by a CNF formula having at most two variables in each conjunct.
- The Boolean function f is called *affine* if it can be represented by a conjunction of affine functions.

We remark that Schaefer's term 1-valid coincides with Post's 1-reproducing.

A set S of Boolean functions is called 0-valid (1-valid, Horn, anti-Horn, affine, bijunctive, resp.) iff every function in S is 0-valid (1-valid, Horn, anti-Horn, affine, bijunctive, resp.).

The satisfiability problem for S-CSPs (S-CSPs with constants, resp.) is denoted by $\mathrm{CSP}(S)$ ($\mathrm{CSP}_C(S)$, resp.). Schaefer's main result, a dichotomy theorem for satisfiability of constraint satisfaction problems (i.e., propositional formulas of the form "conjunction of a set of constraints"), can be stated as follows:

Proposition 5.15. *Let S be a set of Boolean functions. If S is Horn, anti-Horn, affine or bijunctive, then $\mathrm{CSP}_C(S)$ is polynomial-time decideable. In all other cases $\mathrm{CSP}_C(S)$ is $\mathbf{NP}$-complete.*

Proposition 5.16. *Let S be a set of Boolean functions. If S is 0-valid, 1-valid, Horn, anti-Horn, affine or bijunctive, then $\mathrm{CSP}(S)$ is polynomial-time decideable. In all other cases $\mathrm{CSP}(S)$ is $\mathbf{NP}$-complete.*

As a technical tool to obtain the above results, Schaefer defined the set of *existentially quantified S-CSPs with constants*, $\mathrm{Gen}_{\mathrm{C}}(S)$, to be the smallest set of formulas having the following closure properties:

- For any $k \in \mathbb{N}$, any k-ary function $f \in S$, and $u \in (V \cup \{0,1\})^k$, the formula $\tilde{f}(u)$ is in $\mathrm{Gen}_{\mathrm{C}}(S)$, where $\tilde{f}$ is a symbol for f.
- If Φ and Ψ are in $\mathrm{Gen}_{\mathrm{C}}(S)$, then $\Phi \wedge \Psi$ and $(\exists x)\Phi$ (for $x \in V$) are in $\mathrm{Gen}_{\mathrm{C}}(S)$.

Define $\mathrm{Gen}(S) =_{\mathrm{def}} \{\Phi \mid \Phi \in \mathrm{Gen}_{\mathrm{C}}(S) \text{ and } \Phi \text{ has no constants}\}$, $\mathrm{Rep}_{\mathrm{C}}(S) =_{\mathrm{def}} \{f_\Phi \mid \Phi \in \mathrm{Gen}_{\mathrm{C}}(S)\}$ and $\mathrm{Rep}(S) =_{\mathrm{def}} \{f_\Phi \mid \Phi \in \mathrm{Gen}(S)\}$ be the sets of functions represented by formulas from $\mathrm{Gen}_{\mathrm{C}}(S)$ and $\mathrm{Gen}(S)$, resp. The following results proved by Schaefer will be needed in this section:

Proposition 5.17 ([Sch78], Theorem 3.0). *Let S be a set of Boolean functions. If S is Horn, anti-Horn, affine or bijunctive, then* $\mathrm{Rep}_{\mathrm{C}}(S)$ *satisfies the same condition. Otherwise,* $\mathrm{Rep}_{\mathrm{C}}(S)$ *is the set of all Boolean functions* BF.

Proposition 5.18 ([Sch78], Lemma 4.3). *Let S be a set of Boolean functions. Then at least one of the following four statements holds:*

(1) S is 0-valid
(2) S is 1-valid
(3) $f_x, f_{\neg x} \in \mathrm{Rep}(S)$
(4) $f_{x \not\equiv y} \in \mathrm{Rep}(S)$

Finally the following proposition gives an easy possibility to decide whether a given set of Boolean functions S is bijunctive, Horn, anti-Horn or affine. Hence we can easily determine the complexity of $\mathrm{CSP}(S)$ and $\mathrm{CSP}_C(S)$. For this proposition we will use $\wedge$, $\vee$ and $\oplus$ as component wise application of et, vel or aut on k-tuples $t \in \{0,1\}^k$.

Proposition 5.19 ([KK01]). *Let $f\colon \{0,1\}^k \to \{0,1\}$ be a Boolean function. This function is*

- *Horn if and only if for all t_1, t_2 such that $f(t_1) = 1$, $f(t_2) = 1$ also $f(t_1 \wedge t_2) = 1$,*
- *anti-Horn if and only if for all t_1, t_2 such that $f(t_1) = 1$, $f(t_2) = 1$ also $f(t_1 \vee t_2) = 1$,*
- *bijunctive if and only if for all t_1, t_2, t_3 such that $f(t_1) = 1$, $f(t_2) = 1$ and $f(t_3) = 1$ also $f((t_1 \vee t_2) \wedge (t_1 \vee t_3) \wedge (t_2 \vee t_3)) = 1$, and*
- *affine if and only if for all t_1, t_2, t_3 such that $f(t_1) = 1$, $f(t_2) = 1$ and $f(t_3) = 1$ also $f(t_1 \oplus t_2 \oplus t_3) = 1$.*

Note that such criteria were already provided by Schaefer in [Sch78], but Dechter and Pearl [DP92] gave the much easier criterion presented above. Moreover there is an algorithm which produces a defining propositional formula for a given Boolean function that is Horn, anti-Horn, bijunctive or affine (see [DP92, KV98]).

5.4.3 Finding optimal assignments of S-CSPs

In this section, we will consider formulas that are given as a conjunction of constraints (given by Boolean functions applied to a subset of the variables). At first sight, one might hope that the machinery developed by Post and heavily used in the previous sections is applicable here. However, the "upper-level" conjunction is of a restricted nature, and this does not fit into Post's definition of (unrestricted) superposition. Informally, Schaefer-like results cannot be obtained mechanically from Post/Lewis-like results, but new proofs are needed.

Dichotomy theorems for the problem to determine minimal satisfying assignments of constraint satisfaction problems with respect to the component-wise order of Boolean vectors were obtained in [KK99, KK01]. In contrast we will consider lexicographical ordering, as in the previous section.

The main result of this section is to answer the question for what syntactically restricted classes of formulas, given by a set S of Boolean constraints, Proposition 5.2 remains valid. For this, we will consider the following problems:

PROBLEM:	LexMinCSP(S)
INSTANCE:	An S-CSP Φ
OUTPUT:	The lexicographically smallest satisfying assignment of Φ, or "$\perp$" if Φ is unsatisfiable

The corresponding minimization problem for S-CSPs with constants is denoted by LexMinCSP$_C(S)$. We also examine the analogous maximization problems LexMaxCSP(S) and LexMaxCSP$_C(S)$.

There are known algorithms for calculating satisfying assignments of CSPs in polynomial time for certain restricted classes of formulas [CH97]. We first observe that these algorithms can easily be modified to find minimal satisfying assignments.

Lemma 5.20. *Let S be a set of Boolean functions. If S is bijunctive, Horn, anti-Horn or affine, then* $\mathrm{LexMinCSP}_C(C), \mathrm{LexMinCSP}(C) \in \mathbf{FP}$. *If S is 0-valid, then* $\mathrm{LexMinCSP}(S) \in \mathbf{FP}$.

Proof. It is well known (see, e.g., [Pap94]) that in all four cases, the satisfiability problem is in **P**. The claim follows by using the algorithm given in Fig. 5.1. □

We remark that, if S contains at least one function which is not bijunctive, one function which is not Horn, one function which is not anti-Horn, and one function which is not affine, then LexMinCSP$_C(S)$ cannot be in **FP** (unless **P** = **NP**), because Proposition 5.15 shows that the corresponding decision problem (which is the problem of deciding whether there is *any* satisfying assignment, not necessarily the minimal one) is **NP**-complete.

A similar result, which relies on Proposition 5.16, holds for LexMinCSP(S). The only case which requires a bit care is that of a 1-valid set S. Hardness here follows a result in [CH97], which shows that the problem to decide if there exists a satisfying assignment which differs from the vector $(1, 1, \ldots, 1)$ is **NP**-complete.

By the following theorems we strengthen these observations by showing that if LexMinCSP$_C(S)$ or LexMinCSP(S) are not contained in **FP**, then they are already complete for **OptP**.

Theorem 5.21. *Let S be a set of Boolean functions. If S does not fulfill the properties Horn, anti-Horn, bijunctive or affine, then* LexMinCSP$_C(S)$ *is* $\leq^{log}_{1\text{-T}}$*-complete for* **MinP**.

Proof. Obviously LexMinCSP$_C(S) \in$ **MinP**. Now we have to prove $\leq^{log}_{1\text{-T}}$-hardness for **MinP**.

If S does not fulfills the properties Horn, anti-Horn, bijunctive or affine then Proposition 5.17 shows that $\mathrm{Rep}_{\mathrm{C}}(S)$ includes all Boolean functions.

Let f_i be any Boolean function. Proposition 5.17 tells us that there exists an existentially quantified S-CSP $\Psi = \exists y_1 \ldots \exists y_k \Psi'$, representing f_i, where Ψ' contains no quantifier. Any clause of a 3-SAT formula can be represented by one out of a finite number of Boolean functions. So any clause C_i of a 3-SAT formula Φ can be represented by an S-CSP Φ_i. $\mathrm{Var}(\Phi_i)$ consists of the variables in $\mathrm{Var}(C_i)$ plus a number of variables of the form y_j. We pick different sets of y_j-variables for different formulas Φ_i.

Now we construct a function $g_2 \in \mathbf{FL}$ mapping a 3-SAT formula Φ into an S-CSP Ψ by replacing each C_i by the corresponding Φ_i', where $\mathrm{Var}(\Psi)$ consists of $\mathrm{Var}(\Phi) = \{x_1, \ldots, x_n\}$ plus a set of variables of the form y_j. We order the variables by their index and by alphabet, i.e., $x_1 < x_2 < x_3 < \cdots < y_1 < y_2 < \ldots$. Note that we can drop the $\exists$-quantifiers of the variables y_j since we ask for a satisfying assignment of Ψ. The ordering of the variables ensures that in the minimal satisfying assignment of Ψ the variables in $\{x_1, \ldots, x_n\}$ will be minimal with respect to satisfaction of Φ.

Now the function $g_1 \in \mathbf{FL}$ shortens the assignment and removes all bits belonging to the variables y_j. Thus g_1 applied to the minimal satisfying assignment of $\Psi = g_2(\Phi)$ produces the minimal satisfying assignment for Φ. This shows that LexMin3-SAT $\leq^{log}_{1\text{-T}}$ $\mathrm{LexMinCSP}_C(S)$. □

Note that our proof heavily hinges on Schaefer's Proposition 5.17. However Schaefer's technique always introduces new variables, which pose no problem in his context, but are not allowed here. We can only remove these new variables in the end because we have the power of 1-Turing reductions here.

Mainly we are interested in formulas without constants. So we have to get rid of the constants in the construction of the just given proof. This is achieved in the reduction which we now present.

Theorem 5.22. *Let S be a set of Boolean functions. If S is not 0-valid, Horn, anti-Horn, bijunctive or affine, then* $\mathrm{LexMinCSP}(S)$ *is $\leq^{log}_{1\text{-T}}$-complete for* **MinP**.

Proof. Clearly $\mathrm{LexMinCSP}(S) \in \mathbf{MinP}$. We want to show that $\mathrm{LexMinCSP}_C(S)$ reduces to $\mathrm{LexMinCSP}(S)$.

Case 1: S is not 1-valid.

Using Proposition 5.18 we know, that $f_x, f_{\neg x} \in \mathrm{Rep}(S)$ or $f_{x \not\equiv y} \in \mathrm{Rep}(S)$. In what follows, we again sort all variables first alphabetically, and then by index.

Case 1.1: $f_x, f_{\neg x} \in \mathrm{Rep}(S)$.

Let Φ an S-CSP with constants and $\mathrm{Var}(\Phi) = \{x_1, \ldots, x_n\}$. Now we can remove the constants by replacing any 1 by y_1 and 0 by y_0 and adding clauses representing $\{y_1\}$ and $\{\neg y_0\}$. Define the function g_2 such that $g_2(\Phi)$ performs exactly the just described replacement.

Now $I \models_{\min} \Phi$ if and only if $I' =_{\text{def}} I \cup \{y_0 := 0, y_1 := 1\} \models_{\min} \Psi$, where $\Psi =_{\text{def}} g_2(\Phi)$. The function g_1 removes the last two bits (assignments of y_0 and y_1) from I', showing that $\text{LexMinCSP}_C(S) \leq_{1\text{-T}}^{log} \text{LexMinCSP}(S)$.

Case 1.2: $f_{x \not\equiv y} \in \text{Rep}(S)$.

Let Φ be an S-CSP with constants and $\text{Var}(\Phi) = \{x_1, \ldots, x_n\}$. We construct an S-CSP $\Psi =_{\text{def}} \Phi \left[\begin{smallmatrix}0\\v\end{smallmatrix}\right] \left[\begin{smallmatrix}1\\u\end{smallmatrix}\right] \wedge (u \not\equiv v)$ without constants, where u, v are new variables and $v < u < x_1 < x_2 < \cdots < x_n$. Define g_2 by $g_2(\Phi) = \Psi$. If $I_r \models_{\min} \Phi$ then clearly $I' =_{\text{def}} I_r \cup \{v := 0, u := 1\} \models_{\min} \Psi$. If Φ is unsatisfiable, then either Ψ is unsatisfiable, too, or $I'' \models_{\min} \Psi$, where $I'' = I_w \cup \{v := 1, u := 0\}$.

A logspace computable function g_1 can distinguish these cases and compute a minimal assignment for Φ or $\perp$, given an assignment for Ψ. The functions g_1 and g_2 show that $\text{LexMinCSP}_C(S) \leq_{1\text{-T}}^{log} \text{LexMinCSP}(S)$.

Case 2: S is 1-valid.

Having an S-CSP with constants we construct one without constants in logspace by g_2 as follows. Let $f \in S$ be a function which is not 0-valid but 1-valid and $\Psi =_{\text{def}} \Phi \left[\begin{smallmatrix}0\\v\end{smallmatrix}\right] \left[\begin{smallmatrix}1\\u\end{smallmatrix}\right] \wedge \tilde{f}(u, \ldots, u)$, where u, v are new variables and $v < u < x_1 < x_2 < \cdots < x_n$. We claim that $I \models_{\min} \Phi$ iff $I \cup \{v := 0, u := 1\} \models_{\min} \Psi$.

First suppose that $I \models_{\min} \Phi$. It is clear from the clause $\tilde{f}(u, \ldots, u)$ that we have to choose $u := 1$. Since we are interested in the lexicographically smallest solution we have to choose $v := 0$ giving us immediately $I \cup \{v := 0, u := 1\} \models_{\min} \Psi$. Now let $I \cup \{v := 0, u := 1\} \models_{\min} \Psi$. Suppose that there exists a satisfying solution I_s for Φ being lexicographically smaller than I. Obviously $I_s \cup \{v := 0, u := 1\}$ is a lexicographically smaller satisfying assignment than $I \cup \{v := 0, u := 1\}$ giving us a contradiction to $I \cup \{v := 0, u := 1\} \models_{\min} \Psi$. The case that Φ is unsatisfiable can be detected as in case 1.2. This shows that $\text{LexMinCSP}_C(S) \leq_{1\text{-T}}^{log} \text{LexMinCSP}(S)$. □

Thus we get dichotomy theorems for finding lexicographically minimal satisfying assignments of CSPs, both for the case of formulas with constants and without constants.

Corollary 5.23 (Dichotomy Theorem for LexMinCSP(·) **with const.).** *Let S be a set of Boolean functions. If S is bijunctive, Horn, anti-Horn or affine, then* $\text{LexMinCSP}_C(S) \in \mathbf{FP}$. *In all other cases* $\text{LexMinCSP}_C(S)$ *is* $\leq_{1\text{-T}}^{log}$*-complete for* **MinP**.

Corollary 5.24 (Dichotomy Theorem for LexMinCSP(·)**).** *Let S be a set of Boolean functions. If S is 0-valid, bijunctive, Horn, anti-Horn or affine, then we have* $\text{LexMinCSP}(S) \in \mathbf{FP}$. *In all other cases* $\text{LexMinCSP}(S)$ *is* $\leq_{1\text{-T}}^{log}$*-complete for* **MinP**.

If we compare the classes of functions in the statements of the above corollaries with those occurring in Schaefer's results (Propositions 5.15 and 5.16), we immediately obtain the following consequence which completely clarifies the connection between decision and optimization problems for constraint satisfaction.

Corollary 5.25. *Let S be a set of Boolean functions.*

1. $\mathrm{CSP}_C(S)$ *is* **NP***-complete if and only if* $\mathrm{LexMinCSP}_C(S)$ *is* **MinP** *complete.*
2. *If* $\mathrm{CSP}(S)$ *is* **NP***-complete then* $\mathrm{LexMinCSP}(S)$ *is* **MinP** *complete.*
3. *If S is a set of Boolean functions which is 1-valid but is not 0-valid, Horn, anti-Horn, bijunctive, or affine, then* $\mathrm{CSP}(S)$ *is in* **P** *but* $\mathrm{LexMinCSP}(S)$ *is* **MinP** *complete.*

Example 5.26. Note that we will use $\wedge$, $\vee$ and $\oplus$ as component wise application of et, vel or aut on tuples $t \in \{0,1\}^3$. Hierarchical SAT is the variant of 3-SAT where only unnegated variables occur and we require that in each clause if either the first or the second variable are satisfied then the third variable is not satisfied, and if the third variable is satisfied then also the first and second variable are satisfied. In our framework this problem is given by $S = \{f^3\}$, where $f(t) = 1$ iff $t \in \{(1,0,0),(0,1,0),(1,1,1)\}$. Clearly S is 1-valid but not 0-valid. Now by Proposition 5.19 we see that S is not Horn, because $(1,0,0) \wedge (0,1,0) = (0,0,0)$ and $f(0,0,0) \neq 1$, not anti-Horn, since $(1,0,0) \vee (0,1,0) = (1,1,0)$ and $f(1,1,0) \neq 1$, not bijunctive, because $((1,0,0) \vee (0,1,0)) \wedge ((1,0,0) \vee (1,1,1)) \wedge ((0,1,0) \vee (1,1,1)) = (1,1,0) \wedge (1,1,1) \wedge (1,1,1) = (1,1,0)$ and $f(1,1,0) \neq 1$ and not affine, because $(1,0,0) \oplus (0,1,0) \oplus (1,1,1) = (0,0,1)$ and $f(0,0,1) \neq 1$. Thus $\mathrm{CSP}(S)$ is in **P** but $\mathrm{LexMinCSP}(S)$ is **MinP** complete.

Results analogous to the above for the problem of finding maximal assignments can be proved:

Theorem 5.27 (Dichotomy Theorem for $\mathrm{LexMaxCSP}(\cdot)$**).** *Let S be a set of Boolean functions.*

1. *If S is bijunctive, Horn, anti-Horn or affine, then* $\mathrm{LexMaxCSP}_C(S) \in \mathbf{FP}$*. In all other cases* $\mathrm{LexMaxCSP}_C(S)$ *is* $\leq^{log}_{1\text{-T}}$*-complete for* **MaxP***.*
2. *If S is 1-valid, bijunctive, Horn, anti-Horn or affine, then* $\mathrm{LexMaxCSP}(S) \in \mathbf{FP}$*. Otherwise* $\mathrm{LexMaxCSP}(S)$ *is* $\leq^{log}_{1\text{-T}}$*-complete for* **MaxP***.*

5.4.4 Completeness results for the class $\mathbf{P}^{\mathbf{NP}}$

Similar as in Subsect. 5.3.2 for B-formulas, we now examine the problem if the largest variable in a lexicographically minimal assignment of a given S-CSP gets the value 1. Let us denote this problem by

PROBLEM:	$\mathrm{OddLexMinCSP}(S)$
INSTANCE:	An S-CSP $\Phi(x_1, \ldots, x_n)$
QUESTION:	Is in the lexicographically smallest satisfying assignment of Φ the variable x_n set to 1?

In the case that S-CSPs with constants are allowed, we denote this by OddLexMin-$\mathrm{CSP}_C(S)$. (In the case of maximization we use the notation $\mathrm{OddLexMaxCSP}(S)$ and $\mathrm{OddLexMaxCSP}_C(S)$.)

Theorem 5.28 (Dichotomy Theorem for $\mathrm{OddLexMinCSP}_C(\cdot)$**).** *Let S be a set of Boolean functions. If S is affine, bijunctive, Horn or anti-Horn, then* $\mathrm{OddLexMinCSP}_C(S) \in \mathbf{P}$. *In all other cases* $\mathrm{OddLexMinCSP}_C(S)$ *is complete for* $\mathbf{P}^{\mathbf{NP}}$ *under many-one reductions.*

Proof. If S is bijunctive, Horn, anti-Horn or affine, then $\mathrm{OddLexMinCSP}_C(S) \in \mathbf{P}$, since we can use the technique of Lemma 5.20 to find the minimal assignment, and then we accept if and only if the truth value 1 is assigned to the largest variable.

In the other cases we reduce OddLexMin3-SAT to $\mathrm{OddLexMinCSP}_C(S)$. In the proof of Theorem 5.21 we showed how to transform an arbitrary formula Φ with $\mathrm{Var}(\Phi) = \{x_1, \ldots, x_n\}$ into an S-CSP at the cost of introducing new variables of the form y_j. We modify this construction as follows: Introduce one more variable z (larger than all the other variables). Transform Φ into Φ' as described in Theorem 5.21. Finally set $\Phi'' = \Phi' \wedge (x_n \equiv z)$. (Observe that the predicate $\equiv$ is in $\mathrm{Rep}_{\mathrm{C}}(S)$, because of Proposition 5.17.) Let I, I', I'' be the minimal satisfying assignments of Φ, Φ' and Φ''. Observe that they all agree on assignments of the variables in $\mathrm{Var}(\Phi)$. Now we have $I(x_n) = I'(x_n) = I''(x_n) = I''(z)$. Thus $\Phi \in$ OddLexMin3-SAT if and only if $\Phi'' \in \mathrm{OddLexMinCSP}_C(S)$, which proves the claimed hardness result. □

Theorem 5.29 (Dichotomy Theorem for $\mathrm{OddLexMinCSP}(\cdot)$**).** *Let S be a set of Boolean functions. If S is 0-valid, bijunctive, Horn, anti-Horn or affine, then* $\mathrm{OddLexMinCSP}(S) \in \mathbf{P}$. *In all other cases* $\mathrm{OddLexMinCSP}(S)$ *is complete for* $\mathbf{P}^{\mathbf{NP}}$ *under many-one reductions.*

Proof. Similar to the proof of the previous theorem. The easy case is obvious. In the hard case define Φ'' as above, and then use the construction of Theorem 5.22 to remove the constants. Let Φ''' be the resulting formula. The variables introduced in this last step should be smaller than z. Then we can argue as in the previous proof that z is assigned one in a minimal assignment for Φ''' if and only if x_n is assigned one in a minimal assignment for Φ. □

Again, results for maximal assignments are proved analogously:

Theorem 5.30 (Dichotomy Theorem for $\mathrm{OddLexMaxCSP}(\cdot)$**).** *Let S be a set of Boolean functions.*

1. *If S is bijunctive, Horn, anti-Horn or affine, then* $\mathrm{OddLexMaxCSP}_C(S) \in \mathbf{P}$. *In all other cases* $\mathrm{OddLexMaxCSP}_C(S)$ *is complete for* $\mathbf{P}^{\mathbf{NP}}$ *under many-one reductions.*
2. *If S is 1-valid, bijunctive, Horn, anti-Horn or affine, then the problem* $\mathrm{OddLexMaxCSP}(S)$ *is in P. In all other cases* $\mathrm{OddLexMaxCSP}(S)$ *is complete for* $\mathbf{P}^{\mathbf{NP}}$ *under many-one reductions.*

In this chapter we determined the complexity of the problem to compute the lexicographically minimal or maximal satisfying assignment of a given propositional formula, and the problem to determine if in this assignment the largest variable is one, for different restricted formula classes. We obtained a number of dichotomy results, showing that the complexity of the first problem is either in **FP** or **OptP**-complete, while the latter problem is either in **P** or $\mathbf{P}^{\mathbf{NP}}$-complete.

One might ask if it is not possible to obtain our results about constraint satisfaction problems from the seemingly more general results obtained in Section 5.3. However this seems not to be the case, because the "upper-level" conjunction is of a restricted nature, and this does not fit into Post's definition of (unrestricted) superposition. Another hint in that direction is that the results we obtain in the constraint satisfaction context do not speak about closed sets of Boolean functions (the Schaefer classes 0-valid, (anti-)Horn, and bijunctive are not closed in the sense of Post).

Finally another question that arises is, if we can even prove our completeness results for many-one reductions (i.e., always have $g_1(x) = x$). However this cannot be expected for "syntactic" reasons. For example, in Sect. 5.3, if we use a non-complete base $\mathrm{S}_1 \subseteq B \subset \mathrm{BF}$ we have to introduce new variables for using them as a replacement for the constants we need to construct our B-formulas. The assignments to these variables have to be removed later, which means that we need the full power of 1-Turing reductions to do some final manipulation of the value of the function we reduce to.

Index

Bibliography

[AT96] M. Agrawal and T. Thierauf. The Boolean isomorphism problem. In 37th *Symposium on Foundation of Computer Science*, pages 422–430. IEEE Computer Society Press, 1996.

[AT00] M. Agrawal and T. Thierauf. The Formula Isomorphism Problem. *SIAM Journal on Computing*, 30(3):990–1009, 2000. cf. [AT96].

[BDG90] J. L. Balcázar, J. Díaz, and J. Gabarró. *Structural Complexity II*. Springer Verlag, Berlin Heidelberg New York, first edition, 1990.

[BDG95] J. L. Balcázar, J. Díaz, and J. Gabarró. *Structural Complexity I*. Springer Verlag, Berlin Heidelberg New York, 2nd edition, 1995.

[BDHM92] G. Buntrock, C. Damm, U. Hertrampf, and C. Meinel. Structure and Importance of Logspace MOD Class. *Mathematical Systems Theory*, 25:223–237, 1992.

[BRS98] B. Borchert, D. Ranjan, and F. Stefan. On the Computational Complexity of Some Classical Equivalence Relations on Boolean Functions. *Theory of Computing Systems*, 31:679–693, 1998.

[Bus87] S. R. Buss. The Boolean formula value problem is in ALOGTIME. In *Proceedings* 19th *Symposium on Theory of Computing*, pages 123–131, 1987.

[CH96] N. Creignou and M. Hermann. Complexity of generalized satisfiability counting problems. *Information and Computation*, 125:1–12, 1996.

[CH97] N. Creignou and J.-J. Hébrard. On generating all solutions of generalized satisfiability problems. *Informatique Théorique et Applications/Theoretical Informatics and Applications*, 31(6):499–511, 1997.

[CKS99] N. Creignou, S. Khanna, and M. Sudan. Complexity Classifications of Boolean Constraint Satisfaction Problems. TR 1999-24, CNRS, University of Caen, 1999.

[Coo71] S. A. Cook. The complexity of theorem proving procedures. In *Proceedings* 3rd *Symposium on Theory of Computation*, pages 151–158, 1971.

[Cre95] N. Creignou. A dichotomy theorem for maximum generalized satisfiability problems. *Journal of Computer and System Sciences*, 51:511–522, 1995.

[DP92] R. Dechter and J. Pearl. Structure identification in relational data. *Artificial Intelligence*, 48:237–270, 1992.

[Fag74] R. Fagin. Generalized first-order spectra and polynomial time recognizable sets. In R. Karp, editor, *Complexity of Computations*, pages 43–73, 1974.

[GHR95] R. Greenlaw, H. J. Hoover, and W. L. Ruzzo. *Limits to Parallel Computation: P-Completeness Theory.* Oxford University Press, New York, 1995.

[Gil77] J. Gill. Computational complexity of probabilistic Turing machines. *SIAM Journal of Computation*, 6:675–695, 1977.

[GJ79] M. R. Garey and D. S. Johnson. *Computers and Intractability.* W. H. Freeman and Company, 1979.

[Gol77] L. M. Goldschlager. The monotone and planar circuit value problems are log-space complete for P. *SIGACT News*, 9:25–29, 1977.

[GP86] L. Goldschlager and I. Parberry. On The Construction Of Parallel Computers From Various Bases Of Boolean Functions. *Theoretical Computer Science*, 43:43–58, 1986.

[HRV00] U. Hertrampf, S. Reith, and H. Vollmer. A note on closure properties of logspace MOD classes. *Information Processing Letters*, 75:91–93, 2000.

[Imm88] N. Immerman. Nondeterministic space is closed under complementation. *SIAM Journal on Computing*, 17(5):935–938, 1988.

[JGK70] S. W. Jablonski, G. P. Gawrilow, and W. B. Kudrajawzew. *Boolesche Funktionen und Postsche Klassen.* Akademie-Verlag, 1970.

[KK99] L. Kirousis and P. G. Kolaitis. Dichotomy theorems for minimal satisfiability. manuscript, 1999.

[KK01] L. M. Kirousis and P. G. Kolaitis. The Complexity of Minimal Satisfiability Problems. In *Proceedings* 18^{th} *Symposium on Theoretical Aspects of Computer Science*, 2001.

[Köb89] J. Köbler. *Strukturelle Komplexität von Anzahlproblemen.* PhD thesis, Universität Stuttgart, Fakultät für Informatik, 1989.

[Kre88] M. W. Krentel. The complexity of optimization functions. *Journal of Computer and System Sciences*, 36:490–509, 1988.

[Kre92] M. W. Krentel. Generalizations of OptP to the polynomial hierarchy. *Theoretical Computer Science*, 97:183–198, 1992.

[KST97] S. Khanna, M. Sudan, and L. Trevisan. Constraint satisfaction: The approximability of minimization problems. In *Proceedings* 12^{th} *Computational Complexity Conference*, pages 282–296. IEEE Computer Society Press, 1997.

[KSW97] S. Khanna, M. Sudan, and D. Williamson. A complete classification of the approximability of maximization problems derived from Boolean constraint satisfaction. In *Proceedings* 29th *Symposium on Theory of Computing*, pages 11–20. ACM Press, 1997.

[KV98] P. G. Kolaitis and M. Y. Vardi. Conjunctive-query containment and constraint satisfaction. In *Proceedings* 17th *ACM Symposium on Principles of Database Systems*, pages 205–213, 1998.

[Lad75] R. Ladner. On the structure of polynomial-time reducibility. *Journal of the ACM*, 22:155–171, 1975.

[Lad77] R. E. Ladner. The circuit value problem is logspace complete for P. *SIGACT News*, pages 18–20, 1977.

[Lev73] L. Levin. Universal sorting problems. *Problems of Information Transmission*, 9:265–266, 1973.

[Lew79] H. R. Lewis. Satisfiability Problems for Propositional Calculi. *Mathematical Systems Theory*, 13:45–53, 1979.

[Pap94] C. H. Papadimitriou. *Computational Complexity.* Addison-Wesley, Reading, MA, 1994.

[Pos41] E. L. Post. Two-value iterative systems, 1941.

[RV00] S. Reith and H. Vollmer. Optimal Satisfiability for Propositional Calculi and Constraint Satisfaction Problems. In *Proceedings* 25th *International Symposium on Mathematical Foundations of Computer Science*, volume 1893 of *Lecture Notes in Computer Science.* Springer Verlag, 2000.

[RW00] S. Reith and K. W. Wagner. The Complexity of Problems Defined by Boolean Circuits. TR 255, Institut für Informatik, Universität Würzburg, 2000. To appear in Proceedings *International Conference Mathematical Foundation of Informatics*, Hanoi, October 25 - 28, 1999.

[Sav70] W. J. Savitch. Relationships between nondeterministic and deterministic tape complexities. *Journal of Computer and System Sciences*, 4:177–192, 1970.

[Sch78] T. J. Schaefer. The complexity of satisfiability problems. In *Proccedings* 10th *Symposium on Theory of Computing*, pages 216–226. ACM Press, 1978.

[Sim75] J. Simon. On Some Central Problems In Computational Complexity. Doctoral Thesis, Dept. of Computer Science, Cornell University, 1975.

[SM73] L. J. Stockmeyer and A. R. Meyer. Word Problems Requiring Exponential Time. In *Proceedings* 5th *Symposium on Theory of Computation*, pages 1–9, 1973.

[Sze87] R. Szelepcsényi. The method of forcing for nondeterministic automata. *Bulletin of the European Association for Theoretical Computer Science*, 33:96–100, 1987.

[Val79] L. G. Valiant. The complexity of computing the permanent. *Theoretical Computer Science*, 8:189–201, 1979.

[Vol94] H. Vollmer. On different reducibility notions for function classes. In *Proceedings* 11^{th} *Symposium on Theoretical Aspects of Computer Science*, volume 775 of *Lecture Notes in Computer Science*, pages 449–460. Springer Verlag, 1994.

[Vol99] H. Vollmer. *Introduction to Circuit Complexity.* Springer, Berlin Heidelberg New York, 1999.

[VW95] H. Vollmer and K. W. Wagner. Complexity classes of optimization functions. *Information and Computation*, 120:198–219, 1995.

[Wag87] K. W. Wagner. More complicated questions about maxima and minima, and some closures of NP. *Theoretical Computer Science*, 51:53–80, 1987.

[Wag94] K. W. Wagner. *Einführung in die Theoretische Informatik.* Springer Verlag, Berlin Heidelberg, 1994.