Boolean Functions and Post's Lattice with Applications to Complexity Theory

E. Böhler, H. Vollmer

Bericht Nr. 290 Januar 2002

Preprint-Reihe
Institut für Informatik
Universität Würzburg

Boolean Functions and Post's Lattice with Applications to Complexity Theory

Lecture Notes for *Logic and Interaction 2002*

Elmar Böhler and Heribert Vollmer

Lehrstuhl für Theoretische Informatik, Universität Würzburg,
Am Hubland, D-97074 Würzburg, Germany,
[boehler,vollmer]@informatik.uni-wuerzburg.de

Introduction

Let B be a class of Boolean functions. By $[B]$ we denote the class of all Boolean functions that can be obtained from functions from B by superposition, i.e., essentially arbitrary composition. That is, $[B]$ consists of all functions of B plus those, that can be obtained by the following rule: If $f(x_1, \ldots, x_n) \in [B]$ and $X_1, \ldots, X_n$ are either Boolean variables or elements from $[B]$, then $f(X_1, \ldots, X_n) \in [B]$. Say that a class B of Boolean functions is *closed* if $[B] = B$.

Note that all functions in $[B]$ can be computed by Boolean circuits with gates from B, or, equivalently, can be defined by propositional formulas with connectives from B. If B contains the projection functions, then $[B]$ is exactly the class of functions computable by circuits over basis B. Closed classes containing the projections are sometimes called *clones*.

In the forties of the preceding century, Emil Post obtained a complete list of all closed classes of Boolean functions, nowadays called *Post's lattice*, and moreover, he proved that each of them has a finite basis and gave a list of bases for all closed classes [Pos41] (see Fig. 1 and Table 1 in Sect. 1 of the present paper). A description of Post's lattice (without a proof of the classification, but a proof of the finite basis theorem) can be found in [Pip97, Chap. 1]. A complete and self-contained presentation of Post's result is given in [JGK70]. Alternative and much shorter proofs can be found in [Ugo88, Zve00].

It is well-known from the famous Cook-Levin-Theorem that the problem to determine if a given propositional formula is satisfiable is NP-complete [Coo71, Lev73]. A question that arises immediately is if there are subclasses of propositional formulas for which the satisfiability problem is easier, e.g., efficiently solvable. One direction one might pursue is to restrict the number of propositional connectives allowed. Instead of the usual formulas over $\{\wedge, \vee, \neg\}$ one might for example restrict oneself to $\{\wedge, \oplus\}$. Since $[B]$ is the class of those functions that can be defined using propositional formulas with allowed propositional connectives taken from B, this question is most naturally studied in connection to Post's lattice: Allowing connectives from B is the same as allowing connectives from $[B]$. Lewis in 1979 [Lew79] obtained the interesting result that the satisfiability of propositional formulas is NP-complete if and only if the connective $x \wedge \neg y$ is allowed (in Post's lattice, this corresponds to the class S_1). Coming back to the above example set $\{\wedge, \oplus\}$, satisfiability turns out to be again NP-complete, since, as we will see, $[\{\wedge, \oplus\}]$ is the class R_0 in Post's lattice, a superclass of S_1.

In the present paper, we first give an introduction to Post's lattice (Sect. 1). Then, we turn to the computational complexity of some problems for propositional formulas with connectives from a set B of Boolean functions, and Boolean circuits with gates for functions in B; among others we study the satisfiability problem and the circuit value problem. Finally, in Sect. 3, we examine Boolean constraint satisfaction problems. Satisfiability of such problems is nothing else than satisfiability for propositional formulas in conjunctive normal form, where the clauses may be constructed using arbitrary Boolean functions instead of disjunctions of literals. The complexity of this type of satisfiability problem was classified, depending on the type of clauses allowed, by Thomas Schaefer [Sch78]. Schaefer's proof is long and complicated – here, we give an alternative proof making use of Post's lattice.

1 Closed Classes of Boolean Functions: Post's Lattice

Boolean circuits and Boolean functions attract and deserve a lot of attention in computer science. We will have a look at superposition, i.e., the mathematical operations that correspond to operations carried out when building Boolean circuits. Whether or not a Boolean function can be described by a Boolean circuit depends solely on the sort of gates one is allowed to use in the construction of the circuit. There are classes of Boolean functions that are closed under superposition: We will discuss these and the result from E. L. Post [Pos41] who identified and characterized each of them.

1.1 Boolean Functions

An *n-ary Boolean function* is a function from $\{0,1\}^n$ to $\{0,1\}$. There are several ways to describe an n-ary Boolean function f. One is to explicitely specify for each n-tuple $a_1, \ldots, a_n$ the function value $f(a_1, \ldots, a_n)$, i.e., to construct a lookup table for all possible inputs of the function, the so called *truth-table*. This form of presentation often is very inconvenient, since the truth table will have exponentially many (exactly 2^n) entries.

Since we have binary function values, every Boolean function also defines an n-ary relation: The set of all n-tuples α with $f(\alpha) = 1$. A second possibility to describe a Boolean function thus is by listing all of these, but again, we will have up to exponentially many tuples.

Much more compact methods to describe Boolean functions are Boolean circuits and propositional formulas, which we will define formally in the next subsection.

Throughout the text, we will refer to some basic Boolean functions (that are often used as gates when building circuits or as connectives when building fomulas), with the notations listed below.

- *0-ary Boolean functions:* $\mathrm{c}_0 =_{\mathrm{def}} 0$ and $\mathrm{c}_1 =_{\mathrm{def}} 1$.
  (We write 0 and 1 in formulas.)
- *1-ary Boolean functions:* $\mathrm{ID}(x) =_{\mathrm{def}} x$; and $\mathrm{NOT}(x) = 1$ iff $x = 0$.
  (In formulas we simply write x for $\mathrm{ID}(x)$ and $\overline{x}$ or $\neg x$ for $\mathrm{NOT}(x)$.)
- *some prominent 2-ary Boolean functions:*

x	y	$\mathrm{AND}(x,y)$	$\mathrm{XOR}(x,y)$	$\mathrm{OR}(x,y)$	$\mathrm{EQ}(x,y)$	$\mathrm{IMP}(x,y)$	$\mathrm{NAND}(x,y)$
0	0	0	0	0	1	1	1
0	1	0	1	1	0	1	1
1	0	0	1	1	0	0	1
1	1	1	0	1	1	1	0
formula notation		$x \wedge y$ or $x \cdot y$ or xy	$x \oplus y$	$x \wedge y$	$x \leftrightarrow y$	$x \rightarrow y$	$x \mid y$

1.2 B-Circuits and Superposition

We will now give a description of B-circuits, where B is a set of Boolean functions. A B circuit is a directed, acyclic graph where each node is labeled either with a variable x_i or a function from B. We will call the nodes of this graph *gates*. The number of edges (also called *wires*) pointing into a gate is called *fan-in*, the number of wires leaving a gate is called *fan-out* of that gate. For reasons that will become clear shortly, we also order the edges pointing into a gate. If a wire leaving gate u is pointing into gate v, we say that u is a *predecessor gate* of v. The gates labeled with a variable must have a fan-in of 0; we call these *input gates*. The fan-in of gates labeled with a Boolean function has to be as large as the arity of that function. We mark one particular non-input gate and call it *output gate*.

The computation of a B-circuit C proceeds as we describe next: Let n be the largest number of an input variable in C. Given an n-bit input string x, every gate in C computes a Boolean value as follows: A gate v labeled with the variable x_i returns the i-the bit in x. A gate v of fan-in k labeled by a Boolean function f computes the value $f(a_1, \dots, a_k)$, where $a_1, \dots, a_k$ are the values computed by the predecessor gates of v, ordered according to the order of those wires connecting v with its predecessors. The value of C on input x is the value computed by the output gate. In this way, C computes an n-ary Boolean function, which we denote by f_C. The class $[B]$ is the set of all functions that can be computed by B-circuits.

There are several differences between "standard" Boolean circuits and B-circuits: In B-circuits the gates must be labeled with functions from B instead of the more familiar $\wedge, \vee, \neg$. Also note that in our definition, the output gate never is an input gate. Standard circuits where the output gate may be an input gate, trivially are able to compute all projections (i.e., all I_k^n with $I_k^n(a_1, \dots, a_n) = a_k$ $(1 \leq k \leq n)$). Here, B circuits not necessarily can compute projections; in fact, for projections to be in $[B]$ we must be able to construct a suitable circuit using B-gates. We will come back to this point later. We remark that for every $[B]$-circuit there is a B-circuit describing the same Boolean function, since every node, labeled with a function from $[B]$, can, per definition, be replaced by an equivalent B-circuit. Finally note, that a propositional formula can be described by a circuit where the fan-out of each node is less or equal to 1. So we treat B-formulas as a special case of (tree-like) B-circuits.

Given a fixed set B, what are the Boolean functions that can be computed by a B-circuit? First of all, surely every f from B is in $[B]$: Just build a circuit that has as many input nodes as f has arguments, and draw an edge from every input node to one additional node, labeled with f.

Now, if we have B-circuits C_1 computing the n-ary Boolean function f_{C_1}, and C_2 computing m-ary f_{C_2}, we can derive new circuits by performing one of the following operations on the circuits:

- We get a new circuit C_1' by just adding one input node to C_1. Since we do not add any new edges, the new node has no influence on the computed Boolean function besides the higher arity. We call this operation *introduction of a fictive variable* and get for all $a_1, \dots, a_{n+1} \in \{0, 1\}$: $f_{C_1'}(a_1, \dots, a_{n+1}) = f_{C_1}(a_1, \dots, a_n)$. In general, we will say a variable of a Boolean function (an input gate of a circuit) is *fictive*, if the value of the function (the circuit) never depends on this variable (this gate).
- Surely if C_1' is derived from C_1 by arbitrarily permuting the input variables, it is again a B-circuit, since in C_1' still solely functions from B are used. This operation will be called *permutation of variables* and if $\pi \in \mathrm{S}_n$ is our permutation, we get for all $a_1, \dots, a_n \in \{0, 1\}$: $f_{C_1'}(a_1, \dots, a_n) = f_{C_1}(a_{\pi^{-1}(1)}, \dots, a_{\pi^{-1}(n)})$.
- Since the fan-out of gates is not restricted whatsoever, we can remove all outgoing edges of one input gate x_i of C_1 and assign them to another input gate x_j. After this operation, x_i is a fictive gate and we drop it. The thus derived circuit C_1' is a B circuit and computes a function of arity $n-1$, given by $f_{C_1'}(a_1, \dots, a_{i-1}, a_{i+1}, \dots, a_n) = f_{C_1}(a_1, \dots, a_{i-1}, a_j, a_{i+1}, \dots, a_n)$. We call this operation *identification of variables*.

- If we replace an input gate x_i of C_1 by the whole circuit C_2, i.e., we replace x_i with the output gate of C_2 and we replace in C_2 every input gate x_j (for $1 \leq j \leq m$) by x_{n-1+j}, we get a new B-circuit C' of arity $n+m-1$. For the Boolean function $f_{C'}$ obtained in this way, we have: $f_{C'}(a_1, \ldots, a_{n-1}, a_n, \ldots, a_{n+m-1}) = f_{C_1}(a_1, \ldots, a_{n-1}, f_{C_2}(a_n, \ldots, a_{n+m-1}))$. This operation is called *substitution*.

We see that we have four operations on Boolean circuits that immediately correspond to operations on Boolen functions. When talking about functions, the operations of introduction of fictive variables, permutation of variables, identification of variables, and substitutution are subsumed under the name *superposition*; thus $[B] = \{ f \mid f$ can be derived from functions from B with a finite number of applications of superposition$\}$. It is interesting to note that superposition is equivalent to arbitrary composition of Boolean functions. So $f \in [B]$ if and only if $f \in B$ or there is a $g \in [B]$ and $X_1, \ldots, X_n$, that are either variables or functions from $[B]$, such that $f = g(X_1, \ldots, X_n)$. We conclude that, if we want to know which Boolean functions can be described by a B-circuit, we do not even need B-circuits at all: It suffices to study the closures of classes of Boolean functions under superposition.

Example 1.1. Let $f(x, y) =_{\mathrm{def}} x \wedge \overline{y}$ be a Boolean function. Is AND in $[\{f\}]$? To answer this question, we have to find a composition of f's that is equal to AND. In this case it is easy to verify that $\mathrm{AND}(x, y) = f(x, f(y, x))$.

1.3 Post's Lattice

We say a set of Boolean functions B is *closed* if $B = [B]$, i.e., no new functions can be derived from compositions of functions from B. Every set $\widehat{B}$ with $[\widehat{B}] = B$ is called a *base* (or *basis*) of B. We also say that $\widehat{B}$ is *complete* in B. As mentioned before, all closed classes were identified by Post [Pos41], who also found a finite basis for each of them (see Table 1). Post also detected the very useful inclusion structure of the classes (see Fig. 1), hence the name *Post's graph* or (as we will shortly prove that the structure is a lattice) *Post's lattice.*

We would like to mention that the names of the classes in Fig. 1 are not those Post used originally. Post was not only interested in closed classes of Boolean functions, but he additionally considered so called "iterative classes" which are missing some of the closure properties of our notion of superposition. This leads to the idiosyncrasy that if we used Post's names, we would have, e.g., classes $\mathrm{P}_1, \mathrm{P}_3, \mathrm{P}_5, \mathrm{P}_6$, but no P_2 and P_4. The terminology used in Fig. 1 was developed by Klaus Wagner in an attempt to construct a consistent scheme of names for closed classes, and further propagated in [RW00, RV00].

We want to introduce some closed classes:

- BF is the class of all Boolean functions.

- For $a \in \{0, 1\}$, a Boolean function f is called *a-reproducing* if $f(a, \ldots, a) = a$. The closed classes R_a contain all a-reproducing Boolean functions.

- For $(a_1, \ldots, a_n), (b_1, \ldots, b_n) \in \{0, 1\}^n$, we say $(a_1, \ldots, a_n) \leq (b_1, \ldots, b_n)$ if $a_i \leq b_i$ for $1 \leq i \leq n$. A Boolean function f is called *monotonic* if for all $\alpha, \beta \in \{0, 1\}^n$ holds: If $\alpha \leq \beta$ then $f(\alpha) \leq f(\beta)$. Typical monotonic Boolean functions are AND and OR. The class of all monotonic Boolean functions is denoted by M.

- A Boolean function f is called *self-dual* if for all $a_1, \ldots, a_n \in \{0, 1\}$ we have $f(a_1, \ldots, a_n) = \neg f(\overline{a_1}, \ldots, \overline{a_n})$. Therefore constants are never self-dual and simple self-dual functions are ID and NOT. The class of all self-dual Boolean functions is called D.

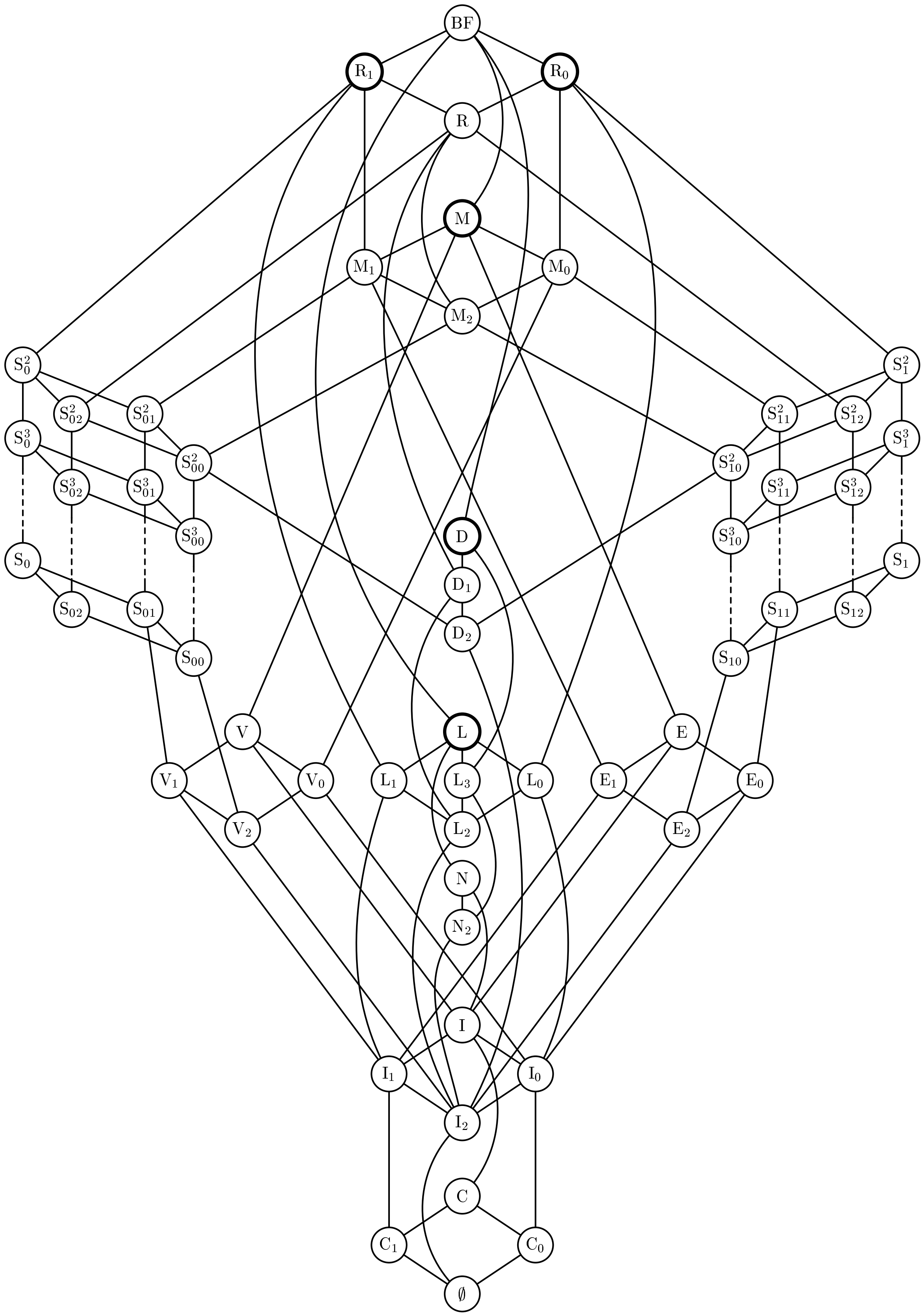

Figure 1: Graph of all closed classes of Boolean functions

Class	Definition	Base(s)
BF	all Boolean functions	$\{\mathrm{AND}, \mathrm{NOT}\}$, $\{\mathrm{OR}, \mathrm{NOT}\}$, $\{\mathrm{NAND}\}$
R_0	$\{ f \in \mathrm{BF} \mid f$ is 0-reproducing $\}$	$\{\mathrm{AND}, \mathrm{XOR}\}$
R_1	$\{ f \in \mathrm{BF} \mid f$ is 1-reproducing $\}$	$\{\mathrm{OR}, x \oplus y \oplus 1\}$
R	$\mathrm{R}_1 \cap \mathrm{R}_0$	$\{\mathrm{OR}, x \wedge (y \oplus z \oplus 1)\}$
M	$\{ f \in \mathrm{BF} \mid f$ is monotonic $\}$	$\{\mathrm{AND}, \mathrm{OR}, \mathrm{c}_0, \mathrm{c}_1\}$
M_1	$\mathrm{M} \cap \mathrm{R}_1$	$\{\mathrm{AND}, \mathrm{OR}, \mathrm{c}_1\}$
M_0	$\mathrm{M} \cap \mathrm{R}_0$	$\{\mathrm{AND}, \mathrm{OR}, \mathrm{c}_0\}$
M_2	$\mathrm{M} \cap \mathrm{R}$	$\{\mathrm{AND}, \mathrm{OR}\}$
S_0^n	$\{ f \in \mathrm{BF} \mid f$ is 0-separating of degree $n \}$	$\{\mathrm{IMP}, \mathrm{dual}(h_n)\}$
S_0	$\{ f \in \mathrm{BF} \mid f$ is 0-separating $\}$	$\{\mathrm{IMP}\}$
S_1^n	$\{ f \in \mathrm{BF} \mid f$ is 1-separating of degree $n \}$	$\{x \wedge \overline{y},\ h_n\}$
S_1	$\{ f \in \mathrm{BF} \mid f$ is 1-separating $\}$	$\{x \wedge \overline{y}\}$
S_{02}^n	$\mathrm{S}_0^n \cap \mathrm{R}$	$\{x \vee (y \wedge \overline{z}), \mathrm{dual}(h_n)\}$
S_{02}	$\mathrm{S}_0 \cap \mathrm{R}$	$\{x \vee (y \wedge \overline{z})\}$
S_{01}^n	$\mathrm{S}_0^n \cap \mathrm{M}$	$\{\mathrm{dual}(h_n), \mathrm{c}_1\}$
S_{01}	$\mathrm{S}_0 \cap \mathrm{M}$	$\{x \vee (y \wedge z), \mathrm{c}_1\}$
S_{00}^n	$\mathrm{S}_0^n \cap \mathrm{R} \cap \mathrm{M}$	$\{x \vee (y \wedge z), \mathrm{dual}(h_n)\}$
S_{00}	$\mathrm{S}_0 \cap \mathrm{R} \cap \mathrm{M}$	$\{x \vee (y \wedge z)\}$
S_{12}^n	$\mathrm{S}_1^n \cap \mathrm{R}$	$\{x \wedge (y, \overline{z}), h_n\}$
S_{12}	$\mathrm{S}_1 \cap \mathrm{R}$	$\{x \wedge (y \vee \overline{z}\}$
S_{11}^n	$\mathrm{S}_1^n \cap \mathrm{M}$	$\{h_n, \mathrm{c}_0\}$
S_{11}	$\mathrm{S}_1 \cap \mathrm{M}$	$\{x \wedge (y \vee z), \mathrm{c}_0\}$
S_{10}^n	$\mathrm{S}_1^n \cap \mathrm{R} \cap \mathrm{M}$	$\{x \wedge (y \vee z), h_n\}$
S_{10}	$\mathrm{S}_1 \cap \mathrm{R} \cap \mathrm{M}$	$\{x \wedge (y \vee z)\}$
D	$\{ f \mid f$ is self-dual $\}$	$\{x\overline{y} \vee x\overline{z} \vee \overline{y} \wedge \overline{z}\}$
D_1	$\mathrm{D} \cap \mathrm{R}$	$\{xy \vee x\overline{z} \vee y\overline{z}\}$
D_2	$\mathrm{D} \cap \mathrm{M}$	$\{xy \vee yz \vee xz\}$
L	$\{ f \mid$ there exists a formula of the form $c_0 \oplus c_1x_1 \oplus \ldots \oplus c_nx_n$ where c_i are constants for all $0 \leq i \leq n$, that describes $f^n \}$	$\{\mathrm{XOR}, \mathrm{c}_1\}$
L_0	$\mathrm{L} \cap \mathrm{R}_0$	$\{\mathrm{XOR}\}$
L_1	$\mathrm{L} \cap \mathrm{R}_1$	$\{\mathrm{EQ}\}$
L_2	$\mathrm{L} \cap \mathrm{R}$	$\{x \oplus y \oplus z\}$
L_3	$\mathrm{L} \cap \mathrm{D}$	$\{x \oplus y \oplus z \oplus \mathrm{c}_1\}$
V	$\{ f \mid$ there exists a formula of the form $c_0 \vee c_1x_1 \vee \ldots \vee c_nx_n$ where c_i are constants for all $0 \leq i \leq n$, that describes $f^n \}$	$\{\mathrm{OR}, \mathrm{c}_0, \mathrm{c}_1\}$
V_0	$[\{\mathrm{OR}\}] \cup [\{\mathrm{c}_0\}]$	$\{\mathrm{OR}, \mathrm{c}_0\}$
V_1	$[\{\mathrm{OR}\}] \cup [\{\mathrm{c}_1\}]$	$\{\mathrm{OR}, \mathrm{c}_1\}$
V_2	$[\{\mathrm{OR}\}]$	$\{\mathrm{OR}\}$
E	$\{ f \mid$ there exists a formula of the form $c_0 \wedge (c_1 \vee x_1) \wedge \ldots \wedge (c_n \vee x_n)$ where c_i are constants for all $0 \leq i \leq n$, that describes $f^n \}$	$\{\mathrm{AND}, \mathrm{c}_0, \mathrm{c}_1\}$
E_0	$[\{\mathrm{AND}\}] \cup [\{\mathrm{c}_0\}]$	$\{\mathrm{AND}, \mathrm{c}_0\}$
E_1	$[\{\mathrm{AND}\}] \cup [\{\mathrm{c}_1\}]$	$\{\mathrm{AND}, \mathrm{c}_1\}$
E_2	$[\{\mathrm{AND}\}]$	$\{\mathrm{AND}\}$
N	$[\{\mathrm{NOT}\}] \cup [\{\mathrm{c}_0\}] \cup [\{\mathrm{c}_1\}]$	$\{\mathrm{NOT}, \mathrm{c}_1\}$, $\{\mathrm{NOT}, \mathrm{c}_0\}$
N_2	$[\{\mathrm{NOT}\}]$	$\{\mathrm{NOT}\}$
I	$[\{\mathrm{ID}\}] \cup [\{\mathrm{c}_1\}] \cup [\{c_0\}]$	$\{\mathrm{ID}, \mathrm{c}_0, \mathrm{c}_1\}$
I_0	$[\{\mathrm{ID}\}] \cup [\{\mathrm{c}_0\}]$	$\{\mathrm{ID}, \mathrm{c}_0\}$
I_1	$[\{\mathrm{ID}\}] \cup [\{\mathrm{c}_1\}]$	$\{\mathrm{ID}, \mathrm{c}_1\}$
I_2	$[\{\mathrm{ID}\}]$	$\{\mathrm{ID}\}$
C	$[\{\mathrm{c}_1\}] \cup [\{\mathrm{c}_0\}]$	$\{\mathrm{c}_1, \mathrm{c}_0\}$
C_0	$[\{\mathrm{c}_0\}]$	$\{\mathrm{c}_0\}$
C_1	$[\{\mathrm{c}_1\}]$	$\{\mathrm{c}_1\}$

Table 1: List of all closed classes of Boolean functions with examples of bases. Here, $h_n =_{\mathrm{def}} \bigvee_{i=1}^{n+1} x_1 \cdots x_{i-1}x_{i+1} \cdots x_{n+1}$ and $\mathrm{dual}(f)(a_1, \ldots, a_n) = \neg f(\overline{a_1}, \ldots, \overline{a_n})$.

- Boolean functions f that can be described with a linear formula are called *linear*, i.e., there exist constants $c_0, \dots, c_n \in \{0, 1\}$ such that $c_0 \oplus c_1 x_1 \oplus \dots \oplus c_n x_n$ describes f. Obviously the typical linear functions are XOR and EQ. The class of all linear Boolean functions is called L.

- Let $T \subseteq \{0, 1\}^n$ and $a \in \{0, 1\}$. We call T *a-separating* if there exists an $i \in \{1, \dots, n\}$ such that for all $(b_1, \dots, b_n) \in T$ holds $b_i = a$. A Boolean function f is called *a-separating* if $f^{-1}(a)$ is a-separating. The function f is called *a-separating of level k* if every $T \subseteq f^{-1}(a)$ with $|T| = k$ is a-separating. The classes of all a-separating functions are called S_a and the classes of a-separating functions of level k are called S_a^k.

- The class E is the class of Boolean functions that can be described by formulas build over $\wedge$, 0 and 1. V is the class of Boolean functions that can be described by formulas build over $\vee$, 0 and 1. ("E" and "V" stand for "et" and "vel", the Latin names of AND and OR.)

- The class C is the class that contains all constant Boolean functions, i.e. if $f \in \mathrm{C}$ then there is some $b \in \{0, 1\}$ such that $f(a_1, \dots, a_n) = b$ for all $a_1, \dots, a_n$.

- The class I_2 contains all projections, i.e., all Boolean functions I_k^n with $I_k^n(a_1, \dots, a_n) = a_k$ for all $a_1, \dots, a_n \in \{0, 1\}$ and I contains all projections and constants. N_2 contains all projections and all negations of projections and N contains N_2 and all constants.

All other classes in Post's graph can be obtained from the above by intersection, as we illustrate next. Let A and B be closed classes and

- let $A \sqcap B$ be the largest closed class that is contained in both A and B, and

- let $A \sqcup B$ be the smallest closed class that contains both A and B.

Note that surely $A \sqcap B \subseteq A \cap B$. On the other hand, $A \cap B$ is a closed class, since every $f \in [A \cap B]$ can be written as a composition of functions from $A \cap B$, and since both A and B are closed under composition of functions, f is in both A and B. We conclude that $A \sqcap B = A \cap B$. Also per definition obviously $[A \cup B] = A \sqcup B$ holds. It is easy to see, that $\sqcap$ and $\sqcup$ are both associative and commutative. Besides that, $A \sqcap (A \sqcup B) = A$ and $A \sqcup (A \sqcap B) = A$ hold, so the closed classes form a lattice.

Post's classes are those classes B in Post's lattice with the properties that $B \subsetneq \mathrm{BF}$ and that for every B' with $B \subsetneq B' \subseteq \mathrm{BF}$ we obtain $[B'] = \mathrm{BF}$. There are five such classes, namely R_0, R_1, M, D, and L – these are circled bold in Fig. 1. This suggests a very convenient test if a given function f (or set B) is complete in the sense that $[f] = \mathrm{BF}$ ($[B] = \mathrm{BF}$): One just has to make sure that f (or B) is not contained in one of Post's classes.

We want to give some examples that illustrate how comfortable live becomes with Post's lattice:

Example 1.2. Let $f = x \wedge \overline{y}$ be the function from Example 1.1 and let us again wonder whether or not AND is in $[\{f\}]$. When we look at Table 1 we see that $\{f\}$ is the base of a class named S_1. Fig. 1 shows us that the class E_2, that is a subset of S_1, has the base $\{\mathrm{AND}\}$. So obviously $\mathrm{AND} \in \mathrm{S}_1 = [\{f\}]$.

Example 1.3. For problems dealing with Boolean circuits often two variations exist: The first one is to allow constants in the circuit and the second one is to not allow them. Suppose we want to discuss a problem where $\{f\}$-circuits are the instances and f is $x \wedge \overline{y}$, again.

For a given $\{f\}$-circuit C, if we exchange some of the input-gates of C with constants 0, we get an $\{f, c_0\}$-circuit. But since $\{c_0\}$ is a base of the closed class $\mathrm{C}_0 \subseteq \mathrm{S}_1 = [\{f\}]$, we can replace each of the c_0's with an $\{f\}$-circuit describing c_0, so the whole circuit becomes an $\{f\}$-circuit again. Thus, problems are as difficult for $\{f, c_0\}$-circuits as they are for $\{f\}$-circuits.

Now, if we exchange some of the input gates of C with constants 1, we get an $\{f, c_1\}$-circuit. Function c_1 is a base of C_1 which is no subset of S_1 so by using the constant 1 we will be able to describe more than just functions from $[\{f\}]$. Which are these? Since $[\{f, c_1\}] = [\mathrm{S}_1 \cup \mathrm{C}_1] = \mathrm{S}_1 \sqcup \mathrm{C}_1$ this is the smallest closed class containing both S_1 and C_1. A look at Fig. 1 shows us that this is the set of all Boolean functions! So problems for $\{f, c_1\}$-circuits are as difficult as they are for $\{\wedge, \vee, \neg\}$-circuits.

We see that those classes in Post's lattice that contain the constant functions are of particular interest. These are exactly the classes $[B \cup \mathrm{C}]$, where B is an arbitrary set of Boolean functions. Making use of the lattice properties, we thus only have to determine all classes $B \sqcup \mathrm{C}$ for all B in Post's lattice. In this way, one immediately obtains Fig. 2, presenting the inclusion structure among all closed classes with constants.

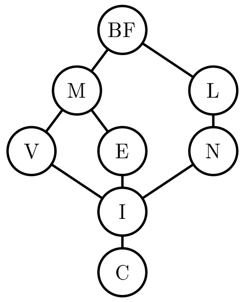

Figure 2: Graph of all closed classes of Boolean functions with constants

Example 1.4. Suppose f is a monotonic Boolean function of arity n, i.e. $f \in \mathrm{M}$. Is f also in L? For this, first note that $\mathrm{L} \not\subseteq \mathrm{M}$, and $\mathrm{M} \cap \mathrm{L} = \mathrm{M} \sqcap \mathrm{L} = \mathrm{I}$ (see Fig. 1). Since $\mathrm{I} = \mathrm{I}_2 \sqcup \mathrm{C} = [\mathrm{I}_2 \cup \mathrm{C}] = \mathrm{I}_2 \cup \mathrm{C}$, f is in L if and only if it is a projection or a constant function. Since f is monotonic, f is constant if and only if $f(0, \dots, 0) = 1$ or $f(1, \dots, 1) = 0$. It is also easy to see (cf. [Böh01]), that f is a projection if and only if there is exactly one $m \in \{1, \dots, n\}$ such that $f(0^{m-1}, 1, 0^{n-m}) = 1$ and $f(1^{m-1}, 0, 1^{m-n}) = 0$, due to the monotonity of f. So by using Post's lattice we are able to find an efficient algorithm deciding whether or not a monotonic Boolean circuit computes a function in L.

Clones, as described in [Pip97] are very similar to closed classes, in fact the two terms refer largely to the same objects. There is one difference: A clone always contains all projections. So the clone generated by B is the set of all Boolean functions that can be build with "standard" B-circuits where input-gates are allowed to be output-gates, or it is the closure of B together with the projections under superposition. The only classes in Fig. 1 that are not clones are $\emptyset$, C_0, C_1, and C.

2 The Complexity of some Problems in Post's Lattice

In computational complexity theory, a large number of problems related to propositional formulas or Boolean circuits have been studied intensively. The most prominent of them is probably the first problem ever shown to be NP-complete [Coo71, Lev73], the problem SAT (see also [GJ79, Problem LO1]), asking if a given propositional formula F is satisfiable. A natural question now is of course, how the complexity of the problem changes if not all propositional formulas but only those using connectives from a previously fixed set B of Boolean functions are allowed. Thus, we are lead to the following problem, first studied systematically by Lewis in 1979 [Lew79]:

Problem:	SAT(B)
Input:	A B-formula F
Question:	Is F satisfiable?

As argued in the previous section, this problem leads immediately to the context of Post's classes. Since these are classes of Boolean functions, it is natural to consider a somewhat more general problem than the above, asking, given a *Boolean circuit* C with gates taken from a set B, if there is an input x such that C on x outputs the value 1. We will denote this problem by $\mathrm{SAT}_c(B)$.

We first present a complete classification of the complexity of $\mathrm{SAT}_c(B)$, showing for which bases B the problem is NP-complete and for which bases it is efficiently solvable. The results says essentially that the problem is NP-complete iff B contains the Boolean function $x \wedge \neg y$.

Theorem 2.1 [RW00]. *If* $[B] \supseteq \mathrm{S}_1$ *then* $\mathrm{SAT}_c(B)$ *is* NP*-complete; in all other cases* $\mathrm{SAT}_c(B)$ *is polynomial-time solvable.*

Proof. Our strategy in this and other proofs in this section is to first try to establish a number of closed classes B, to which NP-completeness for the unrestricted case carries over. Then, making use of Fig. 1, we will examine the hopefully not too many remaining cases.

Looking back at Example 1.3, we know that $[\mathrm{S}_1 \cup \{1\}]$ is the class of all Boolean functions, hence we know that $\mathrm{SAT}_c(\mathrm{S}_1 \cup \{1\})$ is NP-complete. The question now is how to get rid of the constant 1. Since S_1 is a superclass of E it contains the AND-function. Given now an S_1-circuit C, we know that there is another S_1-circuit $\widehat{C}$, equivalent to $C \wedge x$, where x is a new input variable. Moreover, there is an input that makes C output 1 if and only if there is an input that makes $\widehat{C}$ output 1, and additionally, in every such input x is set to 1. Hence, we know that we can use variable x as a replacement for the constant 1. We conclude that for every $(\mathrm{S}_1 \cup \{1\})$-circuit C there is an S_1-circuit $\widehat{C}$ such that there is an input that makes C output 1 if and only if there is an input that makes $\widehat{C}$ output 1. From Example 1.3 and the NP-completeness of SAT, we now conclude that $\mathrm{SAT}_c(\mathrm{S}_1)$ is NP-complete.

Looking at Fig. 1 we see that it only remains to address the classes R_1, M, D, and L. Every R_1-circuit outputs 1 for the input vector consisting only of 1's. Every M-circuit C has the property that the Boolean function it computes is coordinate-wise monotonic; hence there is an input that makes C output 1 iff the all 1's input makes C output 1. Every D-circuit C has the property that either the all 1's input makes C output 1, or, since the Boolean function C computes is self-dual, the all 0's input makes C output 1. Hence, there is an input that makes C output 1. Finally, let C be an L-circuit. We may assume that C consists only of $\oplus$ and 1-gates, since $\{\oplus, 1\}$ is a basis for L. There is an input such that C outputs 1 if and only if the number of paths from the output gate to some constant 1 gate or to some (non-fictive) input gate is odd. This can be checked in polynomial time (in fact, in $\oplus$L). $\square$

The complexity of satisfiability for circuits carries over to the case of formulas, as we observe next.

Corollary 2.2 [Lew79]. *If* $[B] \supseteq \mathrm{S}_1$ *then* $\mathrm{SAT}(B)$ *is* NP*-complete; in all other cases* $\mathrm{SAT}(B)$ *is polynomial-time solvable.*

Proof. Immediately from the above, we conclude that the easy cases carry over to the formula case, i.e., if $[B] \not\supseteq \mathrm{S}_1$ then $\mathrm{SAT}(B)$ is in P. If $[B] \supseteq \mathrm{S}_1$ we would like to proceed as above, transforming an arbitrary propositional formula F into an S_1-formula, but we encounter a problem: The $(\mathrm{S}_1 \cup \{1\})$-formulas for the connectives in F that we need in the transformation may use some input variable more than once, leading to an explosion in formula size when going from F to an equivalent S_1-formula. However, we may assume that F is in conjunctive normal-form with at most 3 literals per clause (3-CNF), since the satisfiability problem for these formulas, denoted by 3SAT, is known to be still NP-complete [GJ79, Problem LO2]. Hence, we insert parentheses in such a way that we get a tree of $\wedge$'s of depth logarithmic in the size of F. Now replacing every $\wedge$ by an equivalent $(\mathrm{S}_1 \cup \{1\})$-formula increases the formula size by only a polynomial in the original size. Thus, 3SAT reduces to $\mathrm{SAT}(\mathrm{S}_1)$, showing that $\mathrm{SAT}(\mathrm{S}_1)$ is NP-complete. $\square$

Coming back to Boolean circuits, a more often looked at problem is not to ask if *there is* any input that makes the circuit output 1, but the problem, *given* an input, to determine the circuit's output. This is the so called *circuit value problem*:

Problem:	$\mathrm{CVP}(B)$
Input:	A B-circuit C and an input vector x
Question:	Does C on input x output 1?

In the case that no restrictions on B are given, this problem is known to be P-complete under logspace-reductions [Lad75] and even under NC^1-reductions, cf. [GHR95, Chap. 6] or [Vol99, Chap. 4.6]. This means that most researchers in the field expect that it allows no efficient parallel solution in the sense of an NC-algorithm, i.e., no polylog-time algorithm using a reasonable amount of hardware (polynomial number of processors); formally the result says that if $\mathrm{P} \neq \mathrm{NC}$ then CVP is not in NC.

The next theorem presents a complete classification of those sets of allowed gates, for which an efficient parallel algorithm for the circuit value problem exists.

Theorem 2.3 [RW00]. *If* $B \subseteq \mathrm{V} \cup \mathrm{L} \cup \mathrm{E}$ *then* $\mathrm{CVP}(B) \in \mathrm{NC}$, *in all other cases* $\mathrm{CVP}(B)$ *is* P*-complete.*

Proof. We will first give efficient (NC-) algorithms in the cases $B \subseteq \mathrm{V} \cup \mathrm{L} \cup \mathrm{E}$. Making use of Post's lattice, we then examine all remaining cases and prove that the circuit value problem is P-complete there.

First, we observe that if the base B contains the constant functions 0 or 1, we may simply treat gates for these functions in the same way as inputs to the circuit, set to 0 or 1, resp. Thus, in general, $\mathrm{CVP}(B \cup \{0,1\})$ is of the same complexity as $\mathrm{CVP}(B)$.

If $B \subseteq \mathrm{V} \cup \mathrm{L} \cup \mathrm{E}$, we thus only have to look at $\{\vee\}$-circuits, $\{\oplus\}$-circuits, and $\{\wedge\}$-circuits. The output of a $\{\vee\}$-circuit is 1 iff there is one path leading back from the output to a 1-input. This is a graph accessibility problem, hence $\mathrm{CVP}(\mathrm{V}) \in \mathrm{NL}$. If C is a $\{\wedge\}$-circuit, the output is 0 iff there is a path leading back from the output to a 0-input. This leads to the upper bound $\mathrm{coNL} = \mathrm{NL}$. If C is a $\oplus$-circuit, the output is 1 iff the number of paths from the output to input gates with value 1 is odd. This is in $\oplus\mathrm{L}$. Since $\mathrm{NL} \cup \oplus\mathrm{L} \subseteq \mathrm{NC}^2$, the first part of the theorem is proven.

If now $B \not\subseteq \mathrm{V} \cup \mathrm{L} \cup \mathrm{E}$, a look at Fig. 1 reveals $\mathrm{S}_{00} \subseteq [B]$, $\mathrm{S}_{10} \subseteq [B]$, or $\mathrm{D}_2 \subseteq [B]$. Considering the bases of S_{00}, S_{10}, and D_2 (see Table 1) or making use of Fig. 2, we see that $\{\wedge, \vee\} \subseteq [B \cup \{0,1\}]$. Thus, $\mathrm{CVP}(B \cup \{0,1\})$ (and by the above, $\mathrm{CVP}(B)$) are at least as hard as the circuit value problem for $\{\vee, \wedge\}$-circuits. This is the so called *monotone circuit value problem*, known to be P-complete [Gol77], cf. also [GHR95, Chap. 6] or [Vol99, Chap. 4.6]. We conclude that $\mathrm{CVP}(B)$ is P-complete. □

The last problem we discuss in some detail in this section stems from the area of optimization problems. The optimization versions of NP problems form the class OptP, introduced by Krentel [Kre88]. Typical problems in this class are to compute the length of the shortest traveling salesperson round-trip, the size of the largest clique in a graph, etc. Formally, a function f belongs to OptP if there is a nondeterministic polynomial-time Turing machine making an output on each of its paths (these machines are sometimes called *metric* Turing machines), such that $f(x)$ is the minimum (or maximum) among all outputs produced by M in the computation on input x. A reducibility notion that has turned out to be reasonable in connection with optimization functions is that of *metric reducibility*: A function f is *metric reducible* to h ($f \leq_m^{\mathrm{p}} h$) if there exist two functions g_1, g_2 computable in polynomial time such that for all x we have: $f(x) = g_1(h(g_2(x)), x)$. A function f is OptP-complete if it belongs to OptP and every other OptP-function reduces to f.

An optimization problem taken from the context of propositional formulas is to look at minimal (or maximal) satisfying assignments of a given formula. More precisely, given a set of variables V, an assignment of truth values to V can be looked at as a binary string of length $|V|$. The lexicographic ordering of such strings defines an order of assignments. The following problem was shown to be OptP-complete by Krentel:

Problem:	LexMinSAT
Input:	A 3-CNF formula F
Question:	the lexicographically smallest satisfying assignment of F, if F is satisfiable; and "$\bot$" else

We now look at restricted propositional formulas, allowing only connectives from a set B of Boolean functions, i.e., we consider the problem LexMinSAT(B), defined as above, but considering B-formulas F as input.

Theorem 2.4 [RV00]. *If* $B \subseteq \mathrm{M} \cup \mathrm{L}$ *then* LexMinSAT(B) *is polynomial-time solvable, in all other cases* LexMinSAT(B) *is* OptP*-complete.*

Proof. Again, we first turn to the easy cases, and then, making use of Post's lattice, prove hardness for all remaining cases.

The following algorithms obviously computes the lexicographically minimal satisfying assignment of a formula F: Consider all variables x in F 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 $\bot$. If no more variables are to consider, we output the resulting assignment.

It is clear that this algorithm runs in polynomial time, if the occurring satisfiability tests are polynomial-time solvable. If F contains only connectives from $\mathrm{M} \cup \mathrm{L}$, this is the case, as we showed above in Corollary 2.2.

It remains to consider the case $B \not\subseteq \mathrm{M} \cup \mathrm{L}$. Fig. 1 shows that $[B] \supseteq \mathrm{S}_{02}$, $[B] \supseteq \mathrm{S}_{12}$, or $[B] \supseteq \mathrm{D}_1$. We will show that in these cases, the LexMinSAT-problem for unrestricted 3-CNF formulas reduces to LexMinSAT(B). Again, it will be our aim to build formulas for the connectives $\wedge$, $\vee$, and $\neg$. As in the proof of Theorem 2.1, the formulas we obtain will make use of constants. We will then exploit the order of variables in such a way that in the minimal model, the values 0 and 1 are necessarily assigned to particular variables. As in the preceding proofs, we may then use these variables as replacements for the constants in our formulas for the connectives $\wedge$, $\vee$, and $\neg$.

More precisely we show that, if there are B-formulas $E(u,v,x,y)$, $N(u,v,x)$ and $C(u,v,x)$ such that $E(0,1,x,y) \equiv x \wedge y$, $N(0,1,x) \equiv \neg x$, $C(0,0,x) \equiv 0$, $C(0,1,x) \equiv x$ and not $C(1,0,x) \equiv C(1,1,x) \equiv 0$, then LexMinSAT$(B)$ is $\leq^{\mathrm{p}}_{met}$-complete for OptP. This follows from the following:

Given a formula F in 3-CNF, we use E and N to transform it (as in the proof of Corollary 2.2) into a B-formula $F'(u,v,x_1,\ldots,x_n)$ such that $F(x_1,\ldots,x_n) \equiv F'(0,1,x_1,\ldots,x_n)$. Next, we use C to define

$$\widehat{F}(u,v,x_1,\ldots,x_n) =_{\mathrm{def}} C(u,v,F').$$

The variables are ordered by $u < v < x_1 < x_2 < \cdots < x_n$.

Claim 1: I is a minimal model of F iff $I \cup \{u := 0, v := 1\}$ is a minimal model of $\widehat{F}$.

"$\Rightarrow$": Let I be a minimal model of F, in symbols: $I \models_{\min} F$. Any model $I' \models \widehat{F}$ cannot assign $v = u := 0$; thus, if there exists an $I' < I \cup \{u := 0, v := 1\}$, then $I'/\{u,v\} = \{u := 0, v := 1\}$ and $I'/\{x_1,\ldots,x_n\} < I$. But this is a contradiction since $I'/\{x_1,\ldots,x_n\} \models F$ (recall that $\widehat{F}(0,1,x_1,\ldots,x_n) \equiv F(x_1,\ldots,x_n)$).

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

Claim 2: F is unsatisfiable iff the minimal model of $\widehat{F}$ is equal to $\{u := 1, v = x_1 = x_2 = \cdots = x_n := 0\}$ or larger in lexicographic order.

For the proof, we first note that $\widehat{F}$ is satisfiable since $C(1,0,X) \not\equiv 0$ or $C(1,1,X) \not\equiv 0$. Hence there exists some model $I \models \widehat{F}$.

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

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

The metric reduction from the LexMinSAT-problem to our LexMinSAT(B) now works as follows: Given F we compute $\widehat{F}$ as described, then we input $\widehat{F}$ to the LexMinSAT(B)-problem and obtain an assignment I; finally, we output "$\perp$" if I is greater than or equal to $\{u := 1, v = x_1 = x_2 = \cdots = x_n := 0\}$, and in all other cases, we remove the assignments for u and v and output the resulting assignment to the other variables.

All we have to do now to finish the proof is to show how to obtain the required sub-formulas. We consider the different cases for B in turn:

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

$[B] \supseteq \mathrm{S}_{12}$: From Table 1 we know that $g(x,y,z) = x \wedge (y \vee \overline{z})$ is a base for S_{12}. Obviously $g(x,y,1) = x \wedge y$, $g(1,0,x) = \neg x$ and $g(x,y,g(x,y,z)) = x \vee (y \wedge z)$. Hence there exist B-formulas $E(u,v,x,y)$, $N(u,v,x)$ and $C(u,v,z)$ such that $E(0,1,x,y) \equiv x \wedge y$, $N(0,1,x) \equiv \neg x$ and $C(u,v,x) \equiv u \vee (v \wedge x)$. Hence, $C(0,0,x) \equiv 0$, $C(0,1,x) \equiv x$, $C(1,0,x) \equiv 1$ and $C(1,1,x) \equiv 1$.

$[B] \supseteq \mathrm{D}_1$: From Table 1 we know that $g(x,y,z) = xy \vee x\overline{z} \vee y\overline{z}$ is a base for D_1. We obtain $g(x,y,1) = x \wedge y$, $g(0,1,x) = \neg x$ and $g(x,y,g(x,y,z)) = xy \vee xz \vee yz$. Since $[B] \supseteq \mathrm{D}_1$ we know that there exist B-formulas $E(u,v,x,y)$, $N(u,v,x)$ and $C(u,v,x)$ such that $E(0,1,x,y) = x \wedge y$, $N(0,1,x) = \neg x$ and $C(u,v,x) = uv \vee ux \vee vx$. Hence, $C(0,0,x) \equiv 0$, $C(0,1,x) \equiv x$ and $C(1,1,x) \equiv 1$. □

Further Complexity Results

A number of further computational problems were looked at in the Post context. These concern, among others, the question to determine the number of satisfying assignments of a given propositional formula and related threshold questions [RW00], the evaluation of quantified Boolean formulas [RW00], the isomorphism problem for Boolean formulas [Rei01], the question to determine the lexicographically maximal satisfying assignment of a given propositional formula and the question whether in this assignment (or the lexicographically minimal satisfying assignment) a particular variable is set to true [RV00], the question to determine the minimal satisfying assignment of a given propositional formula in coordinate-wise partial order [KK01a]. A good source for many complexity results of problems in Post's lattice is Steffen Reith's doctoral dissertation [Rei01].

An interesting question from a complexity point of view is also to determine, given some Boolean function f, in which class of Post's lattice it falls optimally. For this of course, the way f is given plays an important role. For example, f might be presented as a term over $\{\wedge, \oplus\}$

and one might ask if $f \in \mathrm{S}_1$, i.e., if f can be written as a term over $x \wedge \neg y$. Questions of this kind were studied in detail in [Böh01].

3 Post's Lattice and Constraint Satisfaction Problems

In this section, we will consider formulas that are conjunctions of constraints, where the constraints are given by Boolean relations. Let S be a non-empty finite set of Boolean relations (or *constraints*). Now *S-formulas in the Schaefer sense*, or, *S-CSPs*, are conjunctive propositional formulas consisting of clauses built by using relations from S applied to arbitrary variables. Formally, let $S = \{R_1, R_2, \ldots, R_n\}$ be a set of Boolean relations and V be a set of variables. An *S-CSP* F (over V) is a finite conjunction of clauses $F = C_1 \wedge \ldots \wedge C_k$, where each clause (also called *constraint application*) C_i is of the form $\tilde{R}(x_1, \ldots, x_k)$, $R \in S$, $\tilde{R}$ is the symbol representing R, k is the arity of R, and $x_1, \ldots, x_k \in V$. If $F = C_1 \wedge \ldots \wedge C_k$ is a CSP over V, and I is an assignment with respect to V, then $I \models F$ if I satisfies all clauses C_i. Here, a clause $\tilde{R}(x_1, \ldots, x_k)$ is satisfied, if $(I(x_1), \ldots, I(x_k)) \in R$.

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

- The Boolean relation R is *0-valid* (*1-valid* , resp.) if $(0, \ldots, 0) \in R$ ($(1, \ldots, 1) \in R$, resp.).
- The Boolean relation R is *Horn* (*anti-Horn*, resp.) if R can be represented by a CNF formula having at most one unnegated (negated, resp.) variable in any conjunct.
- A Boolean relation R is *bijunctive* if it can be represented by a CNF formula having at most two variables in each conjunct.
- The Boolean relation R is *affine* if it can be represented by a conjunction of affine relations, i.e., a CNF-formula with $\oplus$-clauses.

We remark that, if we identify Boolean relations with Boolean functions in the obvious way, Schaefer's term 1-valid coincides with Post's 1-reproducing.

A set S of Boolean relations is called 0-valid (1-valid, Horn, anti-Horn, affine, bijunctive, resp.) if every relation in S is 0-valid (1-valid, Horn, anti-Horn, affine, bijunctive, resp.).

There are easy criteria to determine if a given relation is Horn, anti-Horn, bijunctive, or affine.

- A relation R is Horn, if and only if for all vectors $x, y \in R$, the vector obtained by taking coordinate-wise the logical conjunction, in symbols: $x \wedge y$, is in R [DP92], see also [CKS00, Lemma 4.8]. Similarly, the property anti-Horn is characterized by coordinate-wise disjunction.
- A relation R is bijunctive, if and only if for all vectors $x, y, z \in R$, the vector obtained by taking coordinate-wise majority (i.e., the ith coordinate is set to 1 iff in at least two of x, y, z the ith coordinate is 1) is in R [Sch78].
- A relation R is affine, if and only if for all vectors $x, y, z \in R$, the vector obtained by taking coordinate-wise logical exclusive-or ($x \oplus y \oplus z$) is in R [Sch78, CH96], see also [CKS00, Lemma 4.10].

Thus, each of these criteria is given in form of a closure property of the set of all vectors in R, and each involves an operation performed on these vectors coordinate-wise.

We prove the characterization of Horn: Suppose R is Horn. Consider a clause C in the representation of R by a Horn formula, and two assignments I_1, I_2 that satisfy C. If one of I_1, I_2

$$\begin{array}{ccccccc}
 & & f(& f(& f(& & f(\\
x_1 & = & 0 & 1 & 1 & \cdots & 0 \\
x_2 & = & 1 & 1 & 0 & \cdots & 1 \\
\vdots & & \vdots & \vdots & \vdots & & \vdots \\
x_m & = & 1 & 1 & 0 & \cdots & 1 \\
 & &) &) &) & &) \\
 & & \parallel & \parallel & \parallel & & \parallel \\
z & = & 0 & 1 & 0 & \cdots & 0
\end{array}$$

Figure 3: If $x_1, \ldots, x_m$ are in R, then z must be in R.

satisfies one of the negative literals in C, then the same literal is satisfied in $I_1 \wedge I_2$. If both I_1, I_2 satisfy the positive literal in C, then of course this literal is also satisfied by $I_1 \wedge I_2$. Thus we see that if $x, y \in R$ then $x \wedge y \in R$. For the converse, suppose that R is not Horn. Then any CNF representation of R must have a clause C that contains at least two positive literals, say x_i and x_j. Let I_1 be the assignment that satisfies variable x_i plus those variables occurring negated in C and falsifies all other variables, and let I_2 be the assignment that satisfies x_j plus those variables occurring negated in C and falsifies all other variables. Then I_1, I_2 satisfy C, but the coordinate-wise conjunction $I_1 \wedge I_2$ does not satisfy C. Hence we found vectors $x, y \in R$ with the property $x \wedge y \notin R$.

The proof of the characterization of anti-Horn is analogous, and the proof of the characterization of bijunctive is very similar. The proof for affine constraints rests on a classical characterization of affine subspaces in linear algebra and can be found in [CKS00, p. 30].

We now come back to the main topic in this section, the satisfiability problem for S-CSPs, denoted by $\mathrm{CSP}(S)$. Schaefer's main result, a dichotomy theorem for satisfiability of constraint satisfaction problems can be stated as follows:

Theorem 3.1 [Sch78]. *Let S be a set of Boolean constraints. If S is 0-valid, 1-valid, Horn, anti-Horn, affine or bijunctive, then* $\mathrm{CSP}(S)$ *is polynomial-time decidable, in all other cases* $\mathrm{CSP}(S)$ *is* NP*-complete.*

In the sequel of this section, we give a proof of Theorem 3.1 making use of Post's classification of closed classes of Boolean functions. The use of Post's lattice to obtain proofs for dichotomy results for constraint satisfaction is implicit in the papers by Jeavons and his group (see, e.g., [JC95, JCG97, JCC98, JCG99]) and in Dalmau's Ph. D. thesis [Dal00] (and some of the facts below appear in these references explicitly). The proof given here follows an argument presented to the authors in an e-mail correspondence by Phokion Kolaitis.

Central for the argument is the notion of a closure property of a Boolean relation (see, e.g., [JC95, Pip97, Dal00]). Let R be a Boolean relation of arity n and let f be a Boolean function of arity m. For f to be a closure property of R we require that if we apply f coordinate-wise to m vectors in R, the obtained vector is again in R; see Fig. 3. Recalling the above characterizations of Schaefer's classes, we see that, e.g., every relation that is Horn is closed under conjunction. More formally, we say that R is *closed* under f (or R *preserves* f, or f is a *polymorphism* of R), if for all $x_1, \ldots, x_m \in R$, where $x_i = (x_i[1], x_i[2], \ldots, x_i[n])$, we have

$$\Big(f(x_1[1], \cdots, x_m[1]), f(x_1[2], \cdots, x_m[2]), \ldots, f(x_1[n], \cdots, x_m[n])\Big) \in R.$$

Let $\mathrm{Pol}(R)$ be the set of all polymorphisms of R. For a set S of Boolean relations, $\mathrm{Pol}(S)$ is the set of Boolean functions that are polymorphisms of every relation in S.

An easy but central observation for the following is that, for every set S, $\mathrm{Pol}(S)$ is a closed class of Boolean functions, in fact, even a clone. (This holds since immediately by definition, Pol(S) is closed under introduction of fictive variables, permutation and identification of variables and substitution, and moreover, contains the projection functions.)

The proof of Schaefer's dichotomy now works as follows: Let S be a set of constraints. We know that we must find Pol(S) somewhere in Post's lattice. The idea is that, if Pol(S) is "hard", meaning somewhere "high" in the diagram, then S is closed under a lot of functions, i.e., S is very restricted and can consist only of very particular constraints, making CSP(S) easy. On the other hand, if Pol(S) consists only of "easy" functions, this means that essentially there are no restrictions on S and thus, CSP(S) is as hard as general propositional satisfiability, i.e., NP-complete.

That Pol(S) is "easy" will mean that it contains only *essentially unary functions*, see Corollary 3.3 below.

We now take a look at Fig. 1. As we said, Pol(S) is a class in that diagram. We now enter a case distinction, but the question is where to start. One easy observation is that if Pol(S) contains either the constant 0 or the constant 1 function, then S must be 0- or 1-valid, hence satisfiability is trivial. This will be our starting point of the proof:

1. Pol(S) contains the constant 0, i.e., Pol(S) $\supseteq \mathrm{C}_0$. Since, as mentioned, Pol(S) is a clone, we actually must have Pol(S) $\supseteq \mathrm{I}_0$. Then we see that S must contain the all-zero vector, i.e., S is 0-valid. Hence every instance of CSP(S) is satisfiable via the all-0 vector.

2. Pol(S) contains the constant 1, i.e., Pol(S) $\supseteq \mathrm{I}_1$. In this case, S must be 1-valid, and every instance of CSP(S) is satisfiable via the all-1 vector.

The minimal clones in Post's lattice that do not contain a constant function are E_2, V_2, L_2, D_2, I_2, and N_2, see Fig. 1. We consider these in turn.

3. Pol(S) $\supseteq \mathrm{E}_2$, i.e., Pol(S) contains the function $x \wedge y$. This means that every relation $R \in S$ is closed under coordinate-wise $\wedge$, hence R is Horn. Satisfiability for Horn formulas is known to be polynomial-time solvable by the so called "Horn algorithm" (HORNSAT $\in$ P, [Pap94, p. 78-9]).

4. Pol(S) $\supseteq \mathrm{V}_2$, i.e., Pol(S) contains the function $x \vee y$. Analogous to the above, we obtain that R can be represented by an anti-Horn-formula. Again, satisfiability is in P by a variation of the Horn algorithm.

5. Pol(S) $\supseteq \mathrm{L}_2$, i.e., Pol(S) contains in particular its basis (see Table 1): the function $x \oplus y \oplus z$. This means that every constraint in R must be affine. A CSP with affine constraints can be looked at as an equation system over GF[2], hence satisfiability can be tested in polynomial time with the Gaussian elimination algorithm.

6. Pol(S) $\supseteq \mathrm{D}_2$, i.e., Pol(S) contains in particular its basis (see Table 1): the function $xz \wedge yz \wedge xy$, i.e., the majority function of arity 3. This means that every constraint in R must be bijunctive. Satisfiability for 2-CNF formulas is polynomial-time solvable (2SAT is even in NL, [Pap94, p. 184-5]).

The remaining cases are now those of the classes I_2 or N_2. The examination of these clones will require a result from universal algebra. Given a set S of Boolean relations, let us denote by S^+ the set of all projections of relations that can be represented by a CNF with applications of constraints in S [JCG99, Definition 4, Lemma 1, and Notation 3], and let $\overline{\overline{S}}$ be the set of all relations R that are closed under every function in Pol(S) [JCG99, Notation 4].

The following result is due to Jeavons et al.:

Proposition 3.2 [JCG99, Theorem 2]. *For every set S of Boolean constraints, $S^+ = \overline{\overline{S}}$.*

Proof. We outline the main ideas in the proof of Proposition 3.2. For the inclusion $S^+ \subseteq \overline{\overline{S}}$ (which is, in fact, not even needed for our proof of Schaefer's dichotomy), we note that, by definition of polymorphism, if two relations are closed under some function, then so is their intersection, and so are all relations obtained from the given relations by projection or equality selection (a relation R' is obtained from R by the equality selection $i = j$ if R' contains all vectors in R in which the ith and jth entry coincide). Any relation in S^+ can be obtained from relations in S by applications of these three operations, hence, since by definition $S \subseteq \overline{\overline{S}}$, we conclude that also $S^+ \subseteq \overline{\overline{S}}$.

We now turn to the reverse inclusion $\overline{\overline{S}} \subseteq S^+$. Jeavons et al.'s main idea is to introduce a particular constraint application instance, the so called *indicator problem of order m*, whose solutions are exactly the m-ary polymorphism of S. The indicator problem is defined using variables standing for m-tuples of Boolean values, i.e., we have 2^m variables which we will for convenience denote in the following by the tuple they stand for, i.e., a variable is an element of $\{0,1\}^m$. The indicator problem uses the relations of S in the obvious way to define the properties of a polymorphism according to its definition, i.e., for all $R \in S$ of arity n and Boolean vectors $t_1, \ldots, t_m \in R$, the indicator problem has a clause of the form $\tilde{R}(x_1, \ldots, x_n)$, where $x_i = (t_1[i], \ldots, t_m[i])$. Since the definition of the indicator problem directly reflects the definition of polymorphism from the above, we see that the solutions to this constraint satisfaction instance are exactly the closure properties f (of arity m) of S in the following sense: If I is a satisfying assignment of the indicator problem of order m, then the function $f\colon \{0,1\}^m \to \{0,1\}$ is a closure property of S, where for every m-tuple t of Boolean values, $f(t) = 1$ if and only if $I(t) = 1$, i.e., t is set to true in I.

Certainly, the indicator problem as a particular constraint application instance is in S^+. We now interpret the solution set of the indicator problem as a relation, i.e., as a table of all solutions. This relation thus has one entry (tuple) for each closure property of S. Let R be an n-ary relation in $\overline{\overline{S}}$ consisting of m tuples, $R = \{t_1, \ldots, t_m\}$. Next, we claim that R can be obtained as a projection to some n input positions of the solution set of the indicator problem of order m. Let $c_i = (t_1[i], \ldots, t_m[i])$ for $1 \leq i \leq n$. Since c_i is an m-tuple of Boolean values, it appears as a variable in the indicator problem. Let T be the projection of the indicator problem (seen as a relation) to these n variables c_i. Every tuple in T is the result of applying some closure property of S to the sequence $t_1, \ldots, t_m$. Since R is closed under all closure properties of S, this tuple in T is also a tuple in R. Every relation is closed under the m-ary Boolean function returning it's i-th argument (I_i^m), hence this function appears as a solution of the indicator problem, yielding a tuple in T, namely t_i. We conclude $R = T$.

Hence, R can be obtained as a suitable projection of the indicator problem, and thus $R \in S^+$. □

We remark that the result of Jeavons et al. holds for constraint sets S over an arbitrary finite (not necessary Boolean) domain.

We now turn to the remaining two cases for $\mathrm{Pol}(S)$:

7. $\mathrm{Pol}(S) = \mathrm{I}_2$. In this case, $\overline{\overline{S}}$ is the set of all Boolean relations R that are closed under all projection functions. This means that for all $x_1, \ldots, x_m \in R$, where $x_i = (x_i[1], \ldots, x_i[n])$, we have $(I_k^m(x_1[1], \ldots, x_m[1]), I_k^m(x_1[2], \ldots, x_m[2]), \ldots, I_k^m(x_1[n], \ldots, x_m[n])) \in R$, where I_k^m is the m-ary projection to the k-th argument. But obviously, $(I_k^m(x_{1,1}, \ldots, x_{m,1}), I_k^m(x_{1,2}, \ldots, x_{m,2}), \ldots, I_k^m(x_{1,n}, \ldots, x_{m,n})) = (x_k[1], x_k[2], \ldots, x_k[n]) = x_k$, hence we claim nothing more than $x_k \in R$, which holds by assumption. Hence $\overline{\overline{S}}$ is the set of all Boolean relations, and by Proposition 3.2, S^+ is the set of all Boolean relations. Thus, $\mathrm{CVP}(S)$ is as hard as the general satisfiability problem, i.e., NP-complete.

8. $\mathrm{Pol}(S) = \mathrm{N}_2$. In this case, $\overline{\overline{S}}$ is the set of all Boolean relations R that are closed under all projection functions and their negations. In particular, $\overline{\overline{S}}$ contains the ternary relation $R_{\mathrm{NAE}} =_{\mathrm{def}} \{(0,0,1), (0,1,0), (0,1,1), (1,0,0), (1,0,1), (1,1,0)\}$. (It can be shown that in fact,

$\overline{\overline{S}}$ is equal to the set of all so called *complementative* relations, see [CKS00].) Looking at CNF formulas with clauses that are applications of constraint R_{NAE}, we see that we deal with a particular satisfiability problem for 3-CNF formulas where we ask if there is an assignment that in every clause makes one variable false and one variable true (stated otherwise: not all three variables in a clause receive the same truth assignment). This is the so called NOT-ALL-EQUAL-3SAT problem which is known to be NP-complete, see, e.g., [Pap94]. Hence, also in this case, CVP(S) is NP-complete, and the proof of Schaefer's dichotomy theorem is finished.

The only cases for Pol(S) that lead to an NP-complete satisfiability problem are those of I_2 and N_2. A common property of these classes is the following: Say that an n-ary function f is *essentially unary* if it is not constant and depends on only one of its arguments, i.e., there is some $i \leq n$ and some unary non-constant Boolean function f' such that $f(x_1, \ldots, x_i, \ldots, x_n) = f'(x_i)$. Of course, the only possibilities for f' are the unary identity or the unary negation. Thus we obtain the following corollary:

Corollary 3.3 [JC95, JCG99]. *Let S be a set of Boolean constraints. If* Pol(S) *consists of only essentially unary functions, then* CSP(S) *is* NP*-complete, otherwise* CSP(S) *is polynomial-time solvable.*

Further Complexity Results

Also in the context of Boolean constraint satisfaction, not only the satisfiability problem was looked at, but many more problems were classified w.r.t. their computational complexity. Considering different versions of satisfiability, isomorphism, optimization and counting problems, dichotomy theorems for classes as NP, US, MaxSNP, OptP, and #P were obtained [Cre95, CH96, CH97, KS98, Jub99, RV00, KK01b, BHRV01], see also the monograph [CKS00]. Satisfiability and learnability of generalized quantified Boolean formulas was studied by Dalmau [Dal00]. Also, the study of Schaefer's formulas lead to remarkable results about approximability of optimization problems in the constraint satisfaction context [KST97, KSW97, Zwi98].

So far, only Schaefer's original dichotomy theorem for satisfiability has been re-proven making use of Post's lattice. We consider it interesting to give an alternative proof along the same lines of, e.g., the dichotomy theorem for counting the number of satisfying assignments from [CH96].

Acknowledgments. We are grateful to Phokion Kolaitis for sending us his proof of Schaefer's dichotomy (Theorem 3.1) and allowing us to include it here. We thank Steffen Reith for many helpful discussions.

References

[BHRV01] E. Böhler, E. Hemaspaandra, S. Reith, and H. Vollmer. Equivalence problems for Boolean constraint satisfaction. Technical Report 282, Fachbereich Mathematik und Informatik, Universität Würzburg, 2001.

[Böh01] E. Böhler. On the relative complexity of Post's classes. Technical Report 286, Fachbereich Mathematik und Informatik, Universität Würzburg, 2001.

[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.

[CKS00] N. Creignou, S. Khanna, and M. Sudan. *Complexity Classifications of Boolean Constraint Satisfaction Problems.* Monographs on Discrete Applied Mathematics. SIAM, 2000.

[Coo71] S. A. Cook. Characterizations of pushdown machines in terms of time-bounded computers. *Journal of the Association for Computing Machinery*, 18:4–18, 1971.

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

[Dal00] V. Dalmau. *Computational complexity of problems over generalized formulas.* PhD thesis, Department de Llenguatges i Sistemes Informàtica, Universitat Politécnica de Catalunya, 2000.

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

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

[GJ79] M. R. Garey and D. S. Johnson. *Computers and Intractability, A Guide to the Theory of NP-Completeness.* Freeman, New York, 1979.

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

[JC95] P. G. Jeavons and D. A. Cohen. An algebraic characterization of tractable constraints. In *Computing and Combinatorics. First International Conference COCOON'95*, volume 959 of *Lecture Notes in Computer Science*, pages 633–642, Berlin Heidelberg, 1995. Springer-Verlag.

[JCC98] P. G. Jeavons, D. Cohen, and M. C. Cooper. Constraints, consistency and closure. *Artificial Intelligence*, 101:251–265, 1998.

[JCG97] P. G. Jeavons, D. A. Cohen, and M. Gyssens. Closure properties of constraints. *Journal of the ACM*, 44(4):527–548, 1997.

[JCG99] P. G. Jeavons, D. A. Cohen, and M. Gyssens. How to determine the expressive power of constraints. *Constraints*, 4:113–131, 1999.

[JGK70] S. W. Jablonski, G. P. Gawrilow, and W. B. Kudrjawzew. *Boolesche Funktionen und Postsche Klassen*, volume 6 of *Logik und Grundlagen der Mathematik.* Friedr. Vieweg & Sohn and C. F. Winter'sche Verlagsbuchhandlung, Braunschweig and Basel, 1970.

[Jub99] L. Juban. Dichotomy theorem for generalized unique satisfiability problem. In *Proceedings 12th Fundamentals of Computation Theory*, volume 1684 of *Lecture Notes in Computer Science*, pages 327–337. Springer Verlag, 1999.

[KK01a] L. M. Kirousis and P. G. Kolaitis. The complexity of minimal satisfiability in Post's lattice. Unpublished notes, 2001.

[KK01b] L. M. Kirousis and P. G. Kolaitis. The complexity of minimal satisfiability problems. In *Proceedings 18th Symposium on Theoretical Aspects of Computer Science*, volume 2010, pages 407–418. Springer Verlag, 2001.

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

[KS98] D. Kavvadias and M. Sideri. The inverse satisfiability problem. *SIAM Journal of Computing*, 28(1):152–163, 1998.

[KST97] S. Khanna, M. Sudan, and L. Trevisan. Constraint satisfaction: The approximability of minimization problems. In *Proceedings 12th 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.

[Lad75] R. E. Ladner. The circuit value problem is log space complete for P. *SIGACT News*, 7(1):12–20, 1975.

[Lev73] L. A. Levin. Universal sorting problems. *Problemi Peredachi Informatsii*, 9(3):115–116, 1973. English translation: *Problems of Information Transmission*, 9(3):265–266.

[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.

[Pip97] N. Pippenger. *Theories of Computability.* Cambridge University Press, Cambridge, 1997.

[Pos41] E. L. Post. The two-valued iterative systems of mathematical logic. *Annals of Mathematical Studies*, 5:1–122, 1941.

[Rei01] S. Reith. Generalized Satisfiability Problems. Dissertation, Fachbereich Mathematik und Informatik, Universität Würzburg, 2001.

[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*, pages 640–649. Springer Verlag, 2000.

[RW00] S. Reith and K. W. Wagner. The complexity of problems defined by Boolean circuits. Technical Report 255, Institut für Informatik, Universität Würzburg, 2000. To appear in *Proceedings International Conference Mathematical Foundation of Informatics*, Hanoi, Oct. 25–28, 1999.

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

[Ugo88] A. B. Ugolnikov. Closed post classes. *Izvestiya VUZ. Matematika*, 32:79–88, 1988.

[Vol99] H. Vollmer. *Introduction to Circuit Complexity – A Uniform Approach.* Texts in Theoretical Computer Science. Springer Verlag, Berlin Heidelberg, 1999.

[Zve00] I. E. Zverovich. Characterizations of closed classes of Boolean functions in terms of forbidden subfunctions and Post classes. Technical Report 17-2000, Rutgers Center for Operations Research, Rutgers University, 2000.

[Zwi98] U. Zwick. Finding almost-satisfying assignments. In *Proceedings 30th Symposium on Theory of Computing*, pages 551–560. ACM Press, 1998.

Preprint-Reihe
Institut für Informatik
Universität Würzburg

Verantwortlich: Die Vorstände des Institutes für Informatik.

[221] T. Peichl, H. Vollmer. *Finite Automata with Generalized Acceptance Criteria.* Januar 1999.

[222] H. Vollmer. *Uniform Characterizations of Complexity Classes.* Januar 1999.

[223] U. Ehrenberger, K. Leibnitz. *Impact of Clustered Traffic Distributions in CDMA Radio Network Planning.* März 1999.

[224] H. Vollmer. *Was leistet die Komplexitätstheorie für die Praxis?* März 1999.

[225] R. Kolla, A. Vodopivec, J. Wolff von Gudenberg. *The IAX Architecture: Interval Arithmetic Extension.* April 1999.

[226] H. Schmitz. *Generalized Deterministic Languages and their Automata: A Characterization of Restricted Temporal Logic.* April 1999.

[227] M. Dümmler. *Using Simulation and Genetic Algorithms to Improve Cluster Tool Performance.* Mai 1999.

[228] R. Harris, S. Köhler. *Possibilities for QoS in Existing Internet Routing Protocols.* Mai 1999.

[229] F. Heister, R. Müller. *An approach for the identification of nonlinear, dynamic processes with Kalman Filter-trained recurrent neural structures.* Mai 1999.

[230] C. Glaßer. *The Boolean Hierarchy over Dot-Depth* 1/2. June 1999.

[231] M. Dümmler. *Simulation und Leistungsbewertung von Cluster Tools in der Halbleiterfertigung.* Juli 1999.

[232] C. Glaßer, S. Reith, H. Vollmer. *Approximation Algorithms for Cellular Network Optimization.* Juli 1999.

[233] S. Kosub, K. W. Wagner. *The Boolean Hierarchy of Partitions.* Juli 1999.

[234] B. Kluge, D. Schäfer. *Featurebasierte Lokalisation eines autonomen mobilen Roboters.* Juli 1999.

[235] M. Dümmler, N. Vicari. *A Numerical Analysis of the* $M/D^b/N$ *Queueing System.* Juli 1999.

[236] M. Buck, H. Noltemeier, D. Schäfer. *Practical Strategies for Hypotheses Elimination on the Self-Localization Problem.* August 1999.

[237] S. Köhler, U. Schäfer. *Performance Comparison of different Class-and-Drop treatment of Data and Acknowledgements in DiffServ IP Networks.* August 1999.

[238] N. Vicari, S. Köhler. *Measuring Internet User Traffic Behavior Dependent on Access Speed.* Oktober 1999.

[239] T. Ebert, H. Vollmer. *On the Autoreducibility of Random Sequences.* Oktober 1999.

[240] H. Schmitz. *Boolean Hierarchies inside Dot-Depth One.* Oktober 1999.

[241] U. Hertrampf, S. Reith, H. Vollmer. *A note on closure properties of logspace MOD classes.* Oktober 1999.

[242] P. McKenzie, H. Vollmer, K. W. Wagner. *Arithmetic Circuits and Polynomial Replacement Systems.* November 1999.

[243] C. Glaßer, H. Schmitz. *Languages of Dot-Depth 3/2.* November 1999.

[244] B. Kluge, D. Schäfer. *Featurebasierte Lokalisation eines autonomen mobilen Roboters - Experimentelle Fallstudie.* Dezember 1999.

[245] C. Glaßer. *Consequences of the Existence of Sparse Sets Hard for NP under a Subclass of Truth-Table Reductions.* Januar 2000.

[246] N. Vicari, S. Köhler, J. Charzinski. *The Dependence of Internet User Characteristics on Access Speed.* Januar 2000.

[247] M. Menth. *Carrying Wireless Traffic over IP Using RTP Multiplexing.* Januar 2000.

[248] P. McKenzie, T. Schwentick, D. Therien, H. Vollmer. *The Many Faces of a Translation.* Februar 2000.

[249] F. Wolz, R. Kolla. *Discrete Floorplanning by Multidimensional Pattern Matching.* Februar 2000.

[250] O. Rose, M. Dümmler, A. Schömig. *On the Validity of Approximation Formulae for Machine Downtimes.* Februar 2000.

[251] S. Reith, H. Vollmer. *Optimal Satisfiability for Propositional Calculi and Constraint Satisfaction Problems.* März 2000.

[252] C. Glaßer, S. Reith, H. Vollmer. *The Complexity of Base Station Positioning in Cellular Networks.* März 2000.

[253] F. Klügl, F. Puppe, P. Schwarz und H. Szczerbicka. *Multiagentsystems and Individual-Based Simulation.* März 2000.

[254] M. Dümmler, O. Rose. *Analysis of the Short Term Impact of Changes in Product Mix.* März 2000.

[255] S. Reith, K. W. Wagner. *The Complexity of Problems Defined by Boolean Circuits.* März 2000.

[256] C. Glaßer, H. Schmitz. *Concatenation Hierarchies and Forbidden Patterns.* März 2000.

[257] S. Kosub. *On NP-Partitions over Posets with an Application to Reducing the Set of Solutions of NP Problems* April 2000.

[258] D. Staehle, S. Köhler, U. Kohlhaas. *Towards an optimization of the routing parameters for IP networks.* Mai 2000.

[259] S. Köhler, M. Menth, N. Vicari. *Analytic Performance Evaluation of the RED Algorithm for QoS in TCP/IP Networks.* Mai 2000.

[260] F. Klügl. *Visuelles Programmieren als Basis einer Modellierumgebung: Eigenschaften, Konzepte, Anforderungen.* Mai 2000.

[261] D. Staehle, K. Leibnitz, P. Tran-Gia. *Source Traffic Modeling of Wireless Applications.* Juni 2000.

[262] L. Berry, S. Köhler, D. Staehle, P. Tran-Gia. *Fast heuristics for optimal routing in large IP networks.* Juli 2000.

[263] M. Menth. *The Performance of Multiplexing Voice and Circuit Switched Data in UMTS over IP-Networks.* Juli 2000.

[264] M. Galota, C. Glaßer, S. Reith, H. Vollmer. *A Polynomial-Time Approximation Scheme for Base Station Positioning in UMTS Networks.* August 2000.

[265] J. Wolff von Gudenberg. *Multimedia Architectures and Interval Arithmetic.* Oktober 2000.

[266] S. Kosub. *Boolean Partitions and Projective Closure.* November 2000.

[267] S. Kosub. *Types of Separability.* November 2000.

[268] K. Leibnitz, A. Krauss. *Performance Evaluation of Interference and Cell Loading in UMTS Networks.* Februar 2001.

[269] S. Reith, H. Vollmer. *Ist P=NP? Einführung in die Theorie der NP-Vollständigkeit.* Februar 2001.

[270] H. Eichelberger, G. Tischler, J. Wolff von Gudenberg. *Comprehensive Graphical Description of the STL.* Februar 2001.

[271] M. Galota, H. Vollmer. *A Generalization of the Büchi-Elgot-Trakhtenbrot Theorem.* Februar 2001.

[272] H. Eichelberger, J. Wolff von Gudenberg. *UMLscript Sprachspezifikation.* Februar 2001.

[273] C. Lautemann, P. McKenzie, T. Schwentick, H. Vollmer. *The Descriptive Complexity Approach to LOGCFL.* Februar 2001.

[274] T. Schwentick, D. Therien, H. Vollmer. *Partially-ordered Two-way Automata: A New Characterization of DA.* März 2001.

[275] V. L. Selivanov. *Relating Automata-theoretic Hierarchies to Complexity-theoretic Hierarchies.* März 2001.

[276] V. L. Selivanov. *Boolean Hierarchy of Partitions over Reducible Bases.* März 2001.

[277] S. Kosub, H. Vollmer. *44. Workshop über Komplexitätstheorie, Datenstrukturen und Effiziente Algorithmen.* Juli 2001.

[278] M. Menth. *A Scalable Protocol Architecture for End-to-End Signaling and Resource Reservation in IP Networks.* Juli 2001.

[279] M. Lerch, G. Tischler, J. Wolff von Gudenberg. *filib++ -Interval Library and Reference Manual.* August 2001.

[280] A. Grigore, A. Heffner, M. Ebner, J. Albert. *Evolution virtueller Pflanzen.* August 2001.

[281] M. Galota, S. Kosub, H. Vollmer. *Generic Separations and Leaf Languages.* September 2001.

[282] E. Böhler, E. Hemaspaandra, S. Reith, H. Vollmer. *Equivalence Problems for Boolean Constraint Satisfaction.* September 2001.

[283] D. Staehle, K. Leibnitz, K. Tsipotis. *QoS of Internet Access with GPRS.* October 2001.

[284] D. Staehle, K. Leibnitz, K. Heck, B. Schröder, A. Weller, P. Tran-Gia. *Analytical Characterization of the Soft Handover Gain in UMTS.* October 2001.

[285] S. Hieß, D. Schäfer. *Taskbasierte Positionsverfolgung und Zielsuche autonomer mobiler Roboter.* October 2001.

[286] E. Böhler. *On the Relative Complexity of Post's Classes.* November 2001.

[287] M. Menth, N. Hauck. *A Graph Theoretical Concept for LSP Hierarchies.* November 2001.

[288] M. Menth, R. Martin. *Performance Evaluation of the Extensions for Control Message Retransmissions in RSVP.* November 2001.

[289] F. Wolz, R. Kolla. *Zu Routingflexibilität und Optimierung konfigurierbarer Multiplexer-Netzwerke.* December 2001.

[290] E. Böhler, H. Vollmer. *Boolean Functions and Post's Lattice with Applications to Complexity Theory.* January 2002.