Equivalence Problems for Boolean Constraint Satisfaction

E. Böhler, E. Hemaspaandra, S. Reith,
H. Vollmer

Bericht Nr. 282 September 2001

Preprint-Reihe
Institut für Informatik
Universität Würzburg

Equivalence Problems for Boolean Constraint Satisfaction*

Elmar Böhler† *Edith Hemaspaandra*‡ *Steffen Reith*§ *Heribert Vollmer*¶

Abstract

A Boolean constraint satisfaction instance is a conjunction of constraint applications, where the allowed constraints are drawn from a fixed set $\mathcal{C}$ of Boolean functions. We consider the problem of determining whether two given constraint satisfaction instances are equivalent in the sense that they possess the same sets of satisfying assignments. We prove a *Dichotomy Theorem* by showing that for all sets $\mathcal{C}$ of allowed constraints, this problem is either polynomial-time solvable or coNP-complete, and we give a simple criterion to determine which case holds.

Another equivalence problem addressed in this paper is the isomorphism problem, the problem of determining whether there exists a renaming of the variables that makes two given constraint satisfaction instances equivalent in the above sense. We prove that this problem is coNP-hard if the corresponding equivalence problem is coNP-hard, and polynomial-time many-one reducible to the graph isomorphism problem in all other cases.

Keywords: propositional logic, constraint satisfaction problems, computational complexity, logic in computer science

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 $\mathcal{C}$ of Boolean functions allowed when constructing formulas. A $\mathcal{C}$*-formula* in his sense now is a conjunction of clauses, where each clause consists of a Boolean function from $\mathcal{C}$ 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 $\mathcal{C}$-formulas hence provides a mathematical model for the examination of the complexity of constraint satisfaction problems, studied in artificial intelligence and database theory. Let CSP($\mathcal{C}$) denote the problem to decide for a given $\mathcal{C}$-formula if it is satisfiable. Schaefer showed that, depending on $\mathcal{C}$, the problem CSP($\mathcal{C}$) is either (1)

*Supported in part by grant NSF-INT-9815095/DAAD-315-PPP-gü-ab.

†Theoretische Informatik, Universität Würzburg, Am Hubland, D-97074 Würzburg, Germany, `boehler@informatik.uni-wuerzburg.de`.

‡Department of Computer Science, Rochester Institute of Technology, Rochester, NY 14623, `eh@cs.rit.edu`. Work done in part while visiting Julius-Maximilians-Universität Würzburg.

§ActiveFilm, Rosengasse 10, D-97070 Würzburg, Germany, `streit@streit.cc`. Work done in part while employed at Julius-Maximilians-Universität Würzburg.

¶Theoretische Informatik, Universität Würzburg, Am Hubland, D-97074 Würzburg, Germany, `vollmer@informatik.uni-wuerzburg.de`.

efficiently (i. e., in polynomial time) solvable or (2) NP-complete (see also [GJ79, Problem LO6]); and he gave a simple criterion that allows one to determine whether (1) or (2) holds. Since the complexity of CSP($\mathcal{C}$) is either easy or hard (and not located in one of the – under the assumption P $\neq$ 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, Schaefer's results regained interest among complexity theorists, and constraint satisfaction problems were studied by a number of researchers [Cre95, CH96, CH97, KS98, Jub99, RV00, KK01], see also the monograph [CKS00]. More generally, problems related to $\mathcal{C}$-formulas not necessarily in conjunctive normal form and $\mathcal{C}$-circuits were studied in [RW00, Rei01]. Considering different versions of satisfiability, optimization and counting problems, dichotomy theorems for classes as NP, US, 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, Zwi98].

As mentioned above, constraint satisfaction problems are used as a programming or query language in fields such as artificial intelligence and database theory, and the above complexity results shed light on the difficulty of design of systems in that areas. A problem with immense importance from a practical perspective is that of determining if two sets of constraints express the same state of affairs (that is: are equivalent), i.e, in the applications: if two programs or queries are equivalent, or if a program matches a given specification. Surprisingly, this problem has not yet been looked at from a complexity point of view. In the case of unrestricted propositional formulas, the equivalence problem is easily seen to be complete for coNP. The main result of the present paper (Theorem 2.2) is a complete clarification of the complexity of determining if two given constraint satisfaction instances are equivalent. We consider constraints drawn from a pre-fixed arbitrary finite set $\mathcal{C}$ of Boolean functions and show that for all such $\mathcal{C}$, the considered problem is either (1) solvable in polynomial time, or (2) complete for coNP. As in Schaefer's result, our proof is constructive in the sense that it allows us to easily determine, given $\mathcal{C}$, if (1) or (2) holds.

Besides the immediate practical relevance of the equivalence problem, we also see our results as contributions to the study of two other decision problems: First, the equivalence problem is a "subproblem" of the minimization problem, i.e., the problem to find out, given a set of constraints, if it can equivalently be expressed with a fewer number of constraints. Second, equivalence relates to the isomorphism problem, studied from a theoretical perspective for various mathematical structures. Most prominently, the question if two given (directed or undirected) graphs are isomorphic is one of the few problems in NP neither known to be in P nor known to be NP-complete [KST93]. The most recent news about graph isomorphism are a number of hardness results (e.g., for NL, PL, and DET) given in [Tor00]. Related to our study are the papers [AT00, BRS98] presenting a number of results concerning isomorphism of propositional formulas. In Section 5, we show (Theorem 5.1) that the isomorphism problem for constraint applications is coNP-hard if the corresponding equivalence problem is coNP-hard, and polynomial-time many-one reducible to the just mentioned graph isomorphism problem in all other cases. We also show that for a number of these cases, the isomorphism problem is in fact polynomial-time many-one equivalent to graph isomorphism (Theorem 5.6).

2 Boolean Constraint Satisfaction Problems

We start by formally introducing constraint satisfaction problems. The definitions necessary for the isomorphism problem will be given in Section 5.

Definition 2.1

1. A *constraint* C is a Boolean function from $\{0,1\}^k$ to $\{0,1\}$. k is the *arity* of C.
2. If C is a constraint of arity k, and $x_1, x_2, \ldots, x_k$ are (not necessarily distinct) variables, then $C(x_1, x_2, \ldots, x_k)$ is a *constraint application of* C.
3. If C is a constraint of arity k, and for $1 \leq i \leq k$, x_i is a variable or a constant (0 or 1), then $C(x_1, x_2, \ldots, x_k)$ is a constraint application of C *with constants.*

The decision problems we will examine in this paper are the following.

Definition 2.2 Let $\mathcal{C}$ be a finite set of constraints.

1. CSP($\mathcal{C}$) is the problem of, given a set S of constraint applications of $\mathcal{C}$, to decide whether S is satisfiable, i.e., whether there exists an assignment to the variables of S that satisfies every constraint application in S.
2. $\mathrm{CSP}_c(\mathcal{C})$ is the problem of, given a set S of constraint applications of $\mathcal{C}$ with constants, to decide whether S is satisfiable.
3. EQUIV($\mathcal{C}$) is the problem of, given two sets S and U of constraint applications of $\mathcal{C}$, to decide whether S and U are equivalent, i.e., whether for every assignment to the variables, S is satisfied if and only if U is satisfied.
4. $\mathrm{EQUIV}_c(\mathcal{C})$ is the problem of, given two sets S and U of constraint applications of $\mathcal{C}$ with constants, to decide whether S and U are equivalent.

Clearly, there are an infinite number of CSP($\mathcal{C}$) and EQUIV($\mathcal{C}$) problems. In 1978, Schaefer proved the surprising result that constraint satisfiability problems are either in P or NP-complete. He also completely characterized for which sets of constraints the problem is in P and for which it is NP-complete. Consult the excellent monograph [CKS00] for an almost completely up to date overview of further results and dichotomy theorems for constraint satisfaction problems.

The question of whether satisfiability for CSPs is in P or NP-complete depends on those properties of the involved Boolean functions that we define next.

Definition 2.3 Let C be a constraint.

- C is *0-valid* if $C(\vec{0}) = 1$.
- C is *1-valid* if $C(\vec{1}) = 1$.
- C is *Horn* (a.k.a. weakly negative) if C is equivalent to a CNF formula where each clause has at most one positive variable.
- C is *anti-Horn* (a.k.a. weakly positive) if C is equivalent to a CNF formula where each clause has at most one negative variable.
- C is *bijunctive* if C is equivalent to a 2CNF formula.

- C is *affine* if C is equivalent to a XOR-CNF formula.
- C is *complementive* (a.k.a. C-closed) if for every $s \in \{0,1\}^k$, $C(s) = C(\overline{s})$, where k is the arity of C and $\overline{s} =_{\text{def}} (1 - s_1)(1 - s_2) \cdots (1 - s_k)$ for $s = s_1 s_2 \cdots s_k$.

Let $\mathcal{C}$ be a finite set of constraints. We say $\mathcal{C}$ is 0-valid, 1-valid, Horn, anti-Horn, bijunctive, affine, or complementive if *every* constraint $C \in \mathcal{C}$ is 0-valid, 1-valid, Horn, anti-Horn, bijunctive, affine, or complementive, respectively. Finally, we say that $\mathcal{C}$ is *Schaefer* if $\mathcal{C}$ is Horn or anti-Horn or affine or bijunctive.

Schaefer's theorem can now be stated as follows.

Theorem 2.1 (Schaefer [Sch78]) *Let $\mathcal{C}$ be a finite set of constraints.*

1. *If $\mathcal{C}$ is 0-valid, 1-valid, or Schaefer, then* $\mathrm{CSP}(\mathcal{C})$ *is in* P*; otherwise,* $\mathrm{CSP}(\mathcal{C})$ *is* NP*-complete.*
2. *If $\mathcal{C}$ is Schaefer, then* $\mathrm{CSP}_c(\mathcal{C})$ *is in* P*; otherwise,* $\mathrm{CSP}_c(\mathcal{C})$ *is* NP*-complete.*

In the next two sections of this paper, we consider the problem of determining whether two sets of constraint applications are equivalent. It is immediate that all equivalence problems are in coNP. Note that, in some sense, equivalence is at least as hard as non-satisfiability, since S is not satisfiable if and only if S is equivalent to 0. Thus, we obtain immediately that if $\mathcal{C}$ is not Schaefer, then $\mathrm{EQUIV}_c(\mathcal{C})$ is coNP-complete.

On the other hand, equivalence can be harder than satisfiability. For example, equivalence between Boolean formulas with $\wedge$ and $\vee$ (i.e., without negation) is coNP-complete [EG95] while satisfiability for these formulas is clearly in P. We will prove following dichotomy theorem.

Theorem 2.2 *Let $\mathcal{C}$ be a set of constraints. If $\mathcal{C}$ is Schaefer, then* $\mathrm{EQUIV}(\mathcal{C})$ *and* $\mathrm{EQUIV}_c(\mathcal{C})$ *are in* P*; otherwise,* $\mathrm{EQUIV}(\mathcal{C})$ *and* $\mathrm{EQUIV}_c(\mathcal{C})$ *are* coNP*-complete.*

The rest of the paper is organized as follows. In the next section we will prove that if $\mathcal{C}$ is Schaefer, then $\mathrm{EQUIV}_c(\mathcal{C})$ is in P. In Section 4, we will prove that if $\mathcal{C}$ is not Schaefer, then $\mathrm{EQUIV}(\mathcal{C})$ is coNP-hard. Since it is immediate that constraint equivalence problems are in coNP, this proves our dichotomy theorem. In Section 5, we address the isomorphism problem for CSPs.

3 Equivalence in P

The cases of constraints with polynomial-time equivalence problems are easy to identify.

Theorem 3.1 $\mathrm{EQUIV}_c(\mathcal{C})$ *is truth-table reducible to* $\mathrm{CSP}_c(\mathcal{C})$*.*

Proof. Let S and U be two sets of constraint applications of $\mathcal{C}$ with constants. Note that S and U are equivalent if and only if $U \rightarrow A$ for every constraint application $A \in S$, and $S \rightarrow B$ for every constraint application $B \in U$ (see, e.g., [HK92]). Here and in the following, when we write a set of constraint applications S in a Boolean formula, we take take this to be a shorthand for $\bigwedge_{\widehat{A} \in S} \widehat{A}$.

Given a constraint application $\widehat{A}$ with constants and a set $\widehat{S}$ of constraint applications of $\mathcal{C}$ with constants, it is easy to check whether $\widehat{S} \rightarrow \widehat{A}$ with at most 2^k truth-table queries to $\mathrm{CSP}_c(\mathcal{C})$, where

k is the maximum arity of $\mathcal{C}$: For every assignment to the variables in $\widehat{A}$ that does not satisfy $\widehat{A}$, substitute this partial truth assignment in $\widehat{S}$. $\widehat{S} \rightarrow \widehat{A}$ if and only if none of these substitutions results in a satisfiable set of constraint applications. $\Box$

If $\mathcal{C}$ is Schaefer, then $\mathrm{CSP}_c(\mathcal{C})$ is in P by Schaefer's theorem and we immediately obtain the following corollary.

Corollary 3.2 *If $\mathcal{C}$ is Schaefer, then* $\mathrm{EQUIV}_c(\mathcal{C})$ *is in* P.

4 coNP-hard Equivalence-Problem

Having identified the easy equivalence cases, the following theorem proves the second half of our Dichotomy Theorem 2.2:

Theorem 4.1 *If $\mathcal{C}$ is not Schaefer, then* $\mathrm{EQUIV}(\mathcal{C})$ *is* coNP*-hard.*

First of all, note that this would be easy to prove if we had constants in the language, since for all sets S of constraint applications of $\mathcal{C}$ the following holds: S is not satisfiable if and only if S is equivalent to the constraint 0. Still, we can use this simple observation in the case where $\mathcal{C}$ is not 0-valid and not 1-valid.

Claim 4.2 *If $\mathcal{C}$ is not Schaefer, not 0-valid, and not 1-valid, then* $\mathrm{EQUIV}(\mathcal{C})$ *is* coNP*-hard.*

Proof. We will reduce $\overline{\mathrm{CSP}(\mathcal{C})}$ to $\mathrm{EQUIV}(\mathcal{C})$. Let S be a set of constraint applications of $\mathcal{C}$. As noted above, S is not satisfiable if and only if S is equivalent to 0. Let $C_0 \in \mathcal{C}$ be a constraint that is not 0-valid, and let $C_1 \in \mathcal{C}$ be a constraint that is not 1-valid. Note that, for any variable x, $\{C_0(x,\ldots,x), C_1(x,\ldots,x)\}$ is equivalent to 0, and thus $S \in \overline{\mathrm{CSP}(\mathcal{C})}$ if and only S is equivalent to $\{C_0(y,\ldots,y), C_1(y,\ldots,y)\}$. $\Box$

If $\mathcal{C}$ is not Schaefer, but is 0-valid or 1-valid, then every set of constraint applications of $\mathcal{C}$ is trivially satisfiable (by $\vec{0}$ or $\vec{1}$). In these cases, a reduction from $\overline{\mathrm{CSP}(\mathcal{C})}$ will not help, since $\overline{\mathrm{CSP}(\mathcal{C})}$ is in P. However, we will show that in these cases the problem of determining whether there exists a non-trivial satisfying assignment is NP-complete and we will use the complements of these satisfiability problems to reduce from.

Creignou and Hébrard prove the following result, concerning the existence of non-trivial satisfying assignments ([CH97, Proposition 4.7], their notation for our $\mathrm{CSP}_{\neq\vec{0},\vec{1}}$ is SAT*):

Proposition 4.3 ([CH97]) *If $\mathcal{C}$ is not Schaefer, then* $\mathrm{CSP}_{\neq\vec{0},\vec{1}}(\mathcal{C})$ *is* NP*-complete, where* $\mathrm{CSP}_{\neq\vec{0},\vec{1}}(\mathcal{C})$ *is the problem of, given a set S of constraint applications of $\mathcal{C}$, to decide whether there is a satisfying assignment for S other than $\vec{0}$ and $\vec{1}$.*

$\mathrm{CSP}_{\neq\vec{0},\vec{1}}(\mathcal{C})$ corresponds to the notion of "having a non-trivial satisfying assignment" in the case that $\mathcal{C}$ is 0-valid and 1-valid. We will reduce $\overline{\mathrm{CSP}_{\neq\vec{0},\vec{1}}(\mathcal{C})}$ to $\mathrm{EQUIV}(\mathcal{C})$ in this case in the proof of Claim 4.7 to follow.

For the cases that $\mathcal{C}$ is not 1-valid or not 0-valid, we obtain the following analogues of Proposition 4.3.

Theorem 4.4 *1. If $\mathcal{C}$ is not Schaefer and not 0-valid then* $\mathrm{CSP}_{\neq\vec{1}}(\mathcal{C})$ *is* NP*-complete, where* $\mathrm{CSP}_{\neq\vec{1}}(\mathcal{C})$ *is the problem of, given a set S of constraint applications of $\mathcal{C}$, to decide whether there is a satisfying assignment for S other than $\vec{1}$.*

2. If $\mathcal{C}$ is not Schaefer and not 1-valid then $\mathrm{CSP}_{\neq\vec{0}}(\mathcal{C})$ *is* NP*-complete, where* $\mathrm{CSP}_{\neq\vec{0}}(\mathcal{C})$ *is the problem of, given a set S of constraint applications of $\mathcal{C}$, to decide whether there is a satisfying assignment for S other than $\vec{0}$.*

Proof. Careful inspection of Creignou and Hébrard's proof of Proposition 4.3 shows that following holds if $\mathcal{C}$ is not Schaefer:

1. If $\mathcal{C}$ is not 0-valid and not 1-valid, then $\mathcal{L} = \{S \mid S \in \mathrm{CSP}_{\neq\vec{0},\vec{1}}(\mathcal{C})$ and not $S(\vec{0})$ and not $S(\vec{1})\}$ is NP-complete (this is case 1 of Creignou and Hébrard's proof).
2. If $\mathcal{C}$ is 0-valid and not 1-valid, then $\mathcal{L}_0 = \{S \mid S \in \mathrm{CSP}_{\neq\vec{0},\vec{1}}(\mathcal{C})$ and not $S(\vec{1})\}$ is NP-complete (this is case 2b of Creignou and Hébrard's proof).
3. If $\mathcal{C}$ is 1-valid and not 0-valid, then $\mathcal{L}_1 = \{S \mid S \in \mathrm{CSP}_{\neq\vec{0},\vec{1}}(\mathcal{C})$ and not $S(\vec{0})\}$ is NP-complete (this is case 3b of Creignou and Hébrard's proof).

This almost immediately implies Theorem 4.4. Let $\mathcal{C}$ be not Schaefer and not 0-valid. If $\mathcal{C}$ is not 1-valid, then $\mathcal{L}$ trivially many-one reduces to $\mathrm{CSP}_{\neq\vec{1}}(\mathcal{C})$, since, for S a set of constraint applications of $\mathcal{C}$, $S \in \mathcal{L}$ if and only if not $S(\vec{0})$, not $S(\vec{1})$, and $S \in \mathrm{CSP}_{\neq\vec{1}}(\mathcal{C})$. Similarly, if $\mathcal{C}$ is 1-valid, then $\mathcal{L}_1$ trivially many-one reduces to $\mathrm{CSP}_{\neq\vec{1}}(\mathcal{C})$. This proves part (1) of Theorem 4.4. Part (2) follows by symmetry. □

Claim 4.5 *Let $\mathcal{C}$ be a finite set of constraints.*

1. *If $\mathcal{C}$ is 1-valid, not Schaefer, and not 0-valid, then* $\mathrm{EQUIV}(\mathcal{C})$ *is* coNP*-hard.*
2. *If $\mathcal{C}$ is 0-valid, not Schaefer and not 1-valid, then* $\mathrm{EQUIV}(\mathcal{C})$ *is* coNP*-hard.*

Proof. We will prove the first case; the proof of the second case is similar. We will reduce $\overline{\mathrm{CSP}_{\neq\vec{1}}(\mathcal{C})}$ to $\mathrm{EQUIV}(\mathcal{C})$ as follows. Let S be a set of constraint applications of $\mathcal{C}$ and let $x_1, \ldots, x_n$ be the variables occurring in S. Note that $\vec{1}$ satisfies S, since every constraint in S is 1-valid. Therefore, $S \notin \mathrm{CSP}_{\neq\vec{1}}(\mathcal{C})$ if and only if S is equivalent to $\bigwedge_{i=1}^{n} x_i$. Let $C \in \mathcal{C}$ be not 0-valid. Since C is 1-valid, x_i is equivalent to $C(x_i, \ldots, x_i)$. It follows that $S \notin \mathrm{CSP}_{\neq\vec{1}}(\mathcal{C})$ if and only if S is equivalent to $\{C(x_i, \ldots, x_i) \mid 1 \leq i \leq n\}$. □

The final case is where $\mathcal{C}$ is both 0-valid and 1-valid. We need the following key lemma from Creignou and Hébrard which is used in their proof of Proposition 4.3.

Lemma 4.6 ([CH97], Lemma 4.9(1) rephrased using constraint terminology) *Let $\mathcal{C}$ be a set of constraints that is not Horn, not anti-Horn, not affine, and 0-valid. Then either*

1. *There exists a set V_0 of constraint applications of $\mathcal{C}$ with variables x and y and constant* 0 *such that V_0 is equivalent to $x \rightarrow y$, or*
2. *There exists a set V_0 of constraint applications of $\mathcal{C}$ with variables x, y, z and constant* 0 *such that V_0 is equivalent to $(\overline{x} \wedge \overline{y} \wedge \overline{z}) \vee (x \wedge \overline{y} \wedge z) \vee (\overline{x} \wedge y \wedge z)$.*

Claim 4.7 *Let $\mathcal{C}$ be a finite set of constraints. If $\mathcal{C}$ is not Schaefer but both 0-valid and 1-valid, then* $\mathrm{EQUIV}(\mathcal{C})$ *is* coNP*-hard.*

Proof. We will reduce $\overline{\mathrm{CSP}_{\neq\vec{0},\vec{1}}(\mathcal{C})}$ to $\mathrm{EQUIV}(\mathcal{C})$. Let S be a set of constraint applications of $\mathcal{C}$ and let $x_1, \dots x_n$ be the variables occurring in S. Note that $\vec{0}$ and $\vec{1}$ satisfy S, since every constraint in S is 0-valid and 1-valid. Therefore, $S \notin \mathrm{CSP}_{\neq\vec{0},\vec{1}}(\mathcal{C})$ if and only if S is equivalent to $\bigwedge_{i=1}^{n} x_i \vee \bigwedge_{i=1}^{n} \overline{x_i}$.

First, suppose there is a constraint $C \in \mathcal{C}$ that is non-complementive. (This case is similar to Creignou and Hébrard's case 2a). Let k be the arity of C and let $s \in \{0,1\}^k$ be an assignment such that $C(s) = 1$ and $C(\overline{s}) = 0$. Let $A(x, y)$ be the constraint application $C(a_1, \dots, a_k)$, where $a_i = y$ if $s_i = 1$ and $a_i = x$ if $s_i = 0$. Then $A(0,0) = A(1,1) = 1$, since A is 0-valid and 1-valid; $A(0,1) = 1$, since $C(s) = 1$; and $A(1,0) = 0$, since $C(\overline{s}) = 0$. Thus, $A(x,y)$ is equivalent to $x \to y$. Since $\bigwedge_{i=1}^{n} x_i \vee \bigwedge_{i=1}^{n} \overline{x_i}$ is equivalent to $\bigwedge_{1 \leq i,j \leq n}(x_i \to x_j)$, it follows that $S \notin \mathrm{CSP}_{\neq\vec{0},\vec{1}}(\mathcal{C})$ if and only if S is equivalent to $\bigwedge_{1 \leq i,j \leq n} A(x_i, x_j)$.

It remains to consider the case where every constraint in $\mathcal{C}$ is complementive. Let V_0 be the set of constraint applications of $\mathcal{C}$ with constant 0 from Lemma 4.6. Let V_f be the set of constraint applications of $\mathcal{C}$ that results when we replace each occurrence of 0 in V_0 by f, where f is a new variable. Note that the following holds. There are two cases to consider, depending the form of V_0.

Case 1: $V_0(x,y)$ is equivalent to $(x \to y)$. In this case, consider $V_f(f,x,y)$. Since $V_f(0,x,y)$ is equivalent to $x \to y$, and every constraint in S is complementive, it follows that $V_f(f,x,y)$ is equivalent to $(\overline{f} \wedge (x \to y)) \vee (f \wedge (y \to x))$. Thus, $\bigwedge_{i=1}^{n} x_i \vee \bigwedge_{i=1}^{n} \overline{x_i}$ is equivalent to $\bigwedge_{1 \leq i,j \leq n} V_f(f, x_i, x_j)$, and it follows that $S \notin \mathrm{CSP}_{\neq\vec{0},\vec{1}}(\mathcal{C})$ if and only if S is equivalent to $\bigwedge_{1 \leq i,j \leq n} V_f(f, x_i, x_j)$.

Case 2: $V_0(x,y,z)$ is equivalent to $(\overline{x} \wedge \overline{y} \wedge \overline{z}) \vee (x \wedge \overline{y} \wedge z) \vee (\overline{x} \wedge y \wedge z)$. Since all constraints in V_0 are complementive, $V_f(f,x,y,z)$ behaves as follows: $V_f(0,0,0,0) = V_f(0,1,0,1) = V_f(0,0,1,1) = V_f(1,1,1,1) = V_f(1,0,1,0) = V_f(1,1,0,0) = 1$, and V_f is 0 for all other assignments. Note that $V_f(f,f,x_i,x_j)$ is equivalent to $(x_i \leftrightarrow x_j)$, and thus that $\bigwedge_{i=1}^{n} x_i \vee \bigwedge_{i=1}^{n} \overline{x_i}$ is equivalent to $\bigwedge_{1 \leq i,j \leq n} V_f(f,f,x_i,x_j)$. It follows that $S \notin \mathrm{CSP}_{\neq\vec{0},\vec{1}}(\mathcal{C})$ if and only if S is equivalent to $\bigwedge_{1 \leq i,j \leq n} V_f(f,f,x_i,x_j)$.

□

5 The Isomorphism Problem for Constraint Satisfaction Problems

In this section, we study a somewhat more general problem: The question of whether a set of constraint applications can be made equivalent to a second set of constraint applications using a suitable renaming of its variables. We need some definitions.

Definition 5.1 1. Let $V = \{x_1, \dots, x_n\}$ be a set of variables. By $\pi\colon \{x_1, \dots, x_n\} \to \{x_1, \dots, x_n\}$ we denote a *permutation* of V.

2. Let S be a set of constraint applications. By $\pi(S)$ we denote the set of constraint applications that results when we replace simultaneously all variables x_i of S by $\pi(x_i)$.

3. Let S be a set of constraint applications. We define the number of satisfying assignments of S as $\#_1(S) =_{\rm def} ||\{\, I \mid I$ is an assignment to all variables of S that satisfies every constraint application in $S\,\}||$.

The isomorphism problem now is formally defined as follows:

Definition 5.2 1. $\mathrm{ISO}(\mathcal{C})$ is the problem of, given two sets S and U of constraint applications of $\mathcal{C}$, to decide whether S and U are isomorphic, i.e., whether there exists a permutation π of the variables of S such that $\pi(S)$ is equivalent to U.

2. $\mathrm{ISO}_c(\mathcal{C})$ is the problem of, given two sets S and U of constraint applications of $\mathcal{C}$ with constants, to decide whether S and U are isomorphic.

As was the case for equivalence, isomorphism is in some sense as least as hard as non-satisfiability, since S is not satisfiable if and only if S is isomorphic to 0. Thus, we immediately obtain that if $\mathcal{C}$ is not Schaefer, then $\mathrm{ISO}_c(\mathcal{C})$ is coNP-hard. Unlike the equivalence case however, we do not have a trivial coNP upper bound for isomorphism problems. In fact, there is some evidence [AT00] that the isomorphism problem for CNF formulas is not in coNP. Note that determining whether two formulas or two sets of constraint applications are isomorphic is trivially in Σ_2^p. However, the isomorphism problem for formulas is not Σ_2^p-complete unless the polynomial hierarchy collapses [AT00].

For equivalence, we obtained a polynomial-time upper bound for sets of constraints that are Schaefer. In contrast, it is easy to see that, for example, isomorphism for positive 2CNF formulas (i.e., isomorphism between two sets of constraint applications of $\{(0,1),(1,0),(1,1)\}$) is polynomial-time many-one equivalent to the graph isomorphism problem (GI). This statement will be shown in detail at the end of this section.

The main result of this section is the following theorem.

Theorem 5.1 *Let $\mathcal{C}$ be a set of constraints. If $\mathcal{C}$ is Schaefer, then $\mathrm{ISO}(\mathcal{C})$ and $\mathrm{ISO}_c(\mathcal{C})$ are polynomial-time many-one reducible to* GI*, otherwise,* $\mathrm{ISO}(\mathcal{C})$ *and* $\mathrm{ISO}_c(\mathcal{C})$ *are* coNP*-hard.*

Note that if $\mathcal{C}$ is Schaefer, then the isomorphism problem $\mathrm{ISO}(\mathcal{C})$ cannot be coNP-hard, unless the polynomial-time hierarchy collapses. (This follows since, by our theorem, if $\mathrm{ISO}(\mathcal{C})$ is coNP-hard then GI is coNP-hard, which implies the mentioned collapse as shown in [KST93].) Under the assumption that the polynomial-time hierarchy does not collapse, Theorem 5.1 thus distinguishes an easy case (reducible to GI) and a hard case. In this sense, Theorem 5.1 is again a dichotomy theorem.

The following easy property of the just-defined notions will be central in the proofs of the lower bound part of Theorem 5.1.

Lemma 5.2 *Let S and U be sets of constraint applications of $\mathcal{C}$ with constants. If S is isomorphic to U then $\#_1(U) = \#_1(S)$.*

Proof. First note that every permutation of the variables of S induces a permutation of the rows of the truth-table of S. Now let π be a permutation such that $\pi(S) \equiv U$. Then $\#_1(S) = \#_1(\pi(S))$ and $\#_1(\pi(S)) = \#_1(U)$. □

We will first prove the lower bound part of Theorem 5.1:

Theorem 5.3 *If $\mathcal{C}$ is not Schaefer, then* $\mathrm{ISO}(\mathcal{C})$ *is* coNP*-hard.*

Proof. We first note that a claim analogous to Claim 4.2 also holds for isomorphism, i.e., if $\mathcal{C}$ is not Schaefer, not 0-valid, and not 1-valid, then $\mathrm{ISO}(\mathcal{C})$ is coNP-hard. For the proof, we use the same reduction as in the proof of Claim 4.2 and claim that, again, $S \in \overline{\mathrm{CSP}(\mathcal{C})}$ if and only S is isomorphic to $\{C_0(y,\ldots,y), C_1(y,\ldots,y)\}$. For the direction from left to right note that if $(S, \{C_0(y,\ldots,y), C_1(y,\ldots,y)\}) \in \mathrm{EQUIV}(\mathcal{C})$ then also $(S, \{C_0(y,\ldots,y), C_1(y,\ldots,y)\}) \in \mathrm{ISO}(\mathcal{C})$ by the identity permutation $\pi = \mathrm{id}$. For the other direction note that $S \notin \overline{\mathrm{CSP}(\mathcal{C})}$ iff $\#_1(S) > 0$. Now suppose $(S, \{C_0(y,\ldots,y), C_1(y,\ldots,y)\}) \in \mathrm{ISO}(\mathcal{C})$, then by Lemma 5.2 $\#_1(S) = \#_1(\{C_0(y,\ldots,y), C_1(y,\ldots,y)\})$, which is clearly a contradiction since $\#_1(\{C_0(y,\ldots,y), C_1(y,\ldots,y)\}) = 0$.

Next, we claim, analogously to Claim 4.5, that

(1) if $\mathcal{C}$ is not Schaefer and not 0-valid, then $\mathrm{ISO}(\mathcal{C})$ is coNP-hard; and

(2) If $\mathcal{C}$ is not Schaefer and not 1-valid, then $\mathrm{ISO}(\mathcal{C})$ is coNP-hard.

These are consequences of the following considerations: Certainly, if constraint set S is equivalent to $\{C(x_i,\ldots,x_i) \mid 1 \le i \le n\}$, then $(S, \{C(x_i,\ldots,x_i) \mid 1 \le i \le n\}) \in \mathrm{ISO}(\mathcal{C})$ via $\pi = \mathrm{id}$. For the other direction note that $S \notin \overline{\mathrm{CSP}_{\neq \vec{1}}(\mathcal{C})}$ iff $\#_1(S) \ge 2$, but $\#_1(\{C(x_i,\ldots,x_i) \mid 1 \le i \le n\}) = 1$. By Lemma 5.2 the result follows.

The remaining case is that of a set $\mathcal{C}$ not Schaefer, but both 0-valid and 1-valid. We use the same reduction as in Claim 4.7. Clearly if $<S,U> \in \mathrm{EQUIV}(\mathcal{C})$ then also $<S,U> \in \mathrm{ISO}(\mathcal{C})$ via $\pi = \mathrm{id}$. To show the other direction note that if $S \notin \mathrm{CSP}_{\neq \vec{0},\vec{1}}(\mathcal{C})$ then $\#_1(S) \ge 3$, but $\#_1(\bigwedge_{i=1}^n x_i \vee \bigwedge_{i=1}^n \overline{x_i}) = 2$. Now use Lemma 5.2 to show that S is not isomorphic to one of the formulas constructed in the cases examined in Claim 4.7. This completes the proof of the theorem. $\square$

To complete the proof of Theorem 5.1, it remains to show that if $\mathcal{C}$ is Schaefer, then $\mathrm{ISO}(\mathcal{C})$ and $\mathrm{ISO}_c(\mathcal{C})$ are polynomial-time many-one reducible to GI. We will reduce $\mathrm{ISO}_c(\mathcal{C})$ to graph isomorphism for vertex-colored graphs, a GI variation that is polynomial-time many-one equivalent to GI.

Definition 5.3 VCGI *is the problem of, given two vertex-colored graphs $G_i = (V_i, E_i, \chi_i)$, $i \in \{1,2\}$, $\chi_i : V \to \mathbf{N}$, to determine whether there exists an isomorphism between G_1 and G_2 that preserves colors, i.e., whether there exists a bijection $\phi: V_1 \to V_2$ such that $\{v,w\} \in E_1$ iff $\{\phi(v), \phi(w)\} \in E_2$ and $\chi(v) = \chi(\phi(v))$.*

Proposition 5.4 ([Fon76, BC79]) VCGI *is polynomial-time many-one equivalent to* GI.

By Proposition 5.4, to complete the proof of Theorem 5.1, it suffices to show the following.

Theorem 5.5 *Let $\mathcal{C}$ be a set of constraints. If $\mathcal{C}$ is Schaefer, then* $\mathrm{ISO}_c(\mathcal{C}) \le_m^p$ VCGI.

Proof. Suppose $\mathcal{C}$ is Schaefer, and let S and U be sets of constraint applications of $\mathcal{C}$ with constants. We will first bring S and U into normal form.

Let X be the set of all variables that occur in S or U. Let $\widehat{S}$ be the set of all constraint applications A of $\mathcal{C}$ with constants such that all of A's variables occur in X and such that $S \rightarrow A$. Similarly, let $\widehat{U}$ be the set of all constraint applications B of $\mathcal{C}$ with constants such that all of B's variables occur in X and such that $U \rightarrow B$. It is clear that $S \equiv \widehat{S}$, since $S \subseteq \widehat{S}$ and $S \rightarrow \widehat{S}$. Likewise, $U \equiv \widehat{U}$. Note that $\widehat{S}$ and $\widehat{U}$ are polynomial-time computable (in $|<S,U>|$), since

1. there exist at most $||\mathcal{C}||(||X||+2)^k$ constraint applications A of $\mathcal{C}$ with constants such that all of A's variables occur in X, where k is the maximum arity of constraints in $\mathcal{C}$; and
2. since $\mathcal{C}$ is Schaefer, determining whether $S \rightarrow A$ or $U \rightarrow A$ takes polynomial time, using the same argument as in the proof of Theorem 3.1.

Note that we have indeed brought S and U into normal form, since if $S \equiv U$ then $\widehat{S} = \widehat{U}$. In addition, for any permutation π of X, if $\pi(S) \equiv U$, then $\pi(\widehat{S}) = \widehat{U}$. We remark that this approach of first bringing the sets of constraint applications into normal form is also followed in the $\mathrm{coIP}[1]^{\mathrm{NP}}$ upper bound proof for isomorphism between Boolean formulas [AT00].

It remains to show that we can in polynomial time encode $\widehat{S}$ and $\widehat{U}$ as vertex-colored graphs $G(\widehat{S})$ and $G(\widehat{U})$ such that there exists a permutation π on X with $\pi(\widehat{S}) = \widehat{U}$ if and only if $<G(\widehat{S}), G(\widehat{U})> \in$ VCGI.

For $P = \{A_1(x_{11}, x_{12}, \ldots, x_{1k_1}), A_2(x_{21}, x_{22}, \ldots, x_{2k_2}), \ldots, A_\ell(x_{\ell 1}, x_{\ell 2}, \ldots, x_{\ell k_\ell})\}$ a set of constraint applications of $\mathcal{C}$ with constants and X the set of all variables in P, define $G(P) = (V, E, \chi)$ as the following vertex-colored graph:

- $V = \{0,1\} \cup \{\, p \mid p \in X \,\} \cup \{\, a_{ij} \mid 1 \leq i \leq \ell, 1 \leq j \leq k_i \,\} \cup \{\, A_i \mid 1 \leq i \leq \ell \,\}$. That is, the set of vertices corresponds to the Boolean constants, the variables in X, the constraint applications in P, and the arguments of the constraint applications in P.
- $E = \{\, \{p, a_{ij}\} \mid p = x_{ij} \,\} \cup \{\, \{a_{ij}, A_i\} \mid 1 \leq i \leq \ell, 1 \leq j \leq k_i \,\}$.
- The vertex coloring χ will distinguish the different categories. Of course, we want to allow any permutation of the variables, so we will give all elements of X the same color. In addition, we also need to allow a permutation of constraint applications of the same constraint. Let $\mathcal{C} = \{C_1, \ldots, C_m\}$.
  - $\chi(0) = 0$, $\chi(1) = 1$,
  - $\chi(p) = 2$ for all $p \in X$,
  - $\chi(A_i) = 2 + j$ if $A_i = C_j$, and
  - $\chi(a_{ij}) = 2 + m + j$. (This will ensure that we do not permute the order of the arguments.)

It is straightforward to show that there exists a permutation π on X such that $\pi(\widehat{S}) = \widehat{U}$ if and only if $<G(\widehat{S}), G(\widehat{U})> \in$ VCGI. □

We also show that for a number of simple cases, the isomorphism problem is in fact polynomial-time many-one equivalent to the graph isomorphism problem.

Theorem 5.6 GI *is polynomial-time many-one equivalent to* $\mathrm{ISO}(\{\{(0,1),(1,0),(1,1)\}\})$ *and to* $\mathrm{ISO}_c(\{\{(0,1),(1,0),(1,1)\}\})$.

Note that the constraint $\{(0,1),(1,0),(1,1)\}$ is the binary constraint $x \vee y$, denoted by OR_0 in [CKS99]. Theorem 5.6 can alternatively be formulated as: GI is polynomial-time many-one equivalent the isomorphism problem for positive 2CNF formulas (with or without constants). Also, from [Tor00], we conclude that this isomorphism problem thus is hard for NL, PL, and DET.

Proof. (of Theorem 5.6) It suffices to show that GI $\leq_m^p$ ISO($\{\{(0,1),(1,0),(1,1)\}\}$), since, by Theorem 5.5, ISO($\{\{(0,1),(1,0),(1,1)\}\}$) $\leq_m^p$ GI.

Let $G = <V, E>$ be a graph and let $V = \{1, 2, \ldots, n\}$. We encode G in the obvious way as a formula: $\phi_G = \{\{p_i, p_j\} \mid \{i, j\} \in E\}$. It is immediate that if G and H are two graphs with vertex set $\{1, 2, \ldots, n\}$, then G is isomorphic to H if and only if ϕ_G is isomorphic to ϕ_H. □

From this Theorem 5.6, we conclude that, if we could show that isomorphism for bijunctive, anti-Horn (and, by symmetry, Horn) CSPs is in P, then the graph isomorphism problem is in P, settling a long standing open question.

References

[AT96] M. Agrawal and T. Thierauf. The Boolean isomorphism problem. In 37^{th} *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].

[BC79] K. S. Booth and C. J. Colbourn. Problems polynomially equivalent to graph isomorphism. Technical Report CS-77-01, University of Waterloo, 1979.

[BRS98] B. Borchert, D. Ranjan, and F. Stephan. On the computational complexity of some classical equivalence relations on Boolean functions. *Theory of Computing Systems*, 31:679–693, 1998.

[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. Technical Report 1999-24, CNRS, University of Caen, 1999.

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

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

[EG95] T. Eiter and G. Gottlob. Identifying the minimal transversals of a hypergraph and related problems. *SIAMG Journal on Computing*, 24(6):1278–1304, 1995.

[Fon76] M. Fontet. Automorphismes de graphes et planarité. In *Asterisque*, pages 73–90. 1976.

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

[HK92] P. Hammer and A. Kogan. Horn functions and their DNF. *Information Processing Letters*, 44:23–29, 1992.

[Jub99] L. Juban. Dichotomy theorem for generalized unique satisfiability problem. In *Proceedings* 12^{th} *Fundamentals of Computation Theory*, volume 1684 of *Lecture Notes in Computer Science*, pages 327–337. Springer Verlag, 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*, volume 2010, pages 407–418. Springer Verlag, 2001.

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

[KST93] J. Köbler, U. Schöning, and J. Torán. *The Graph Isomorphism Problem: its Structureal Complexity.* Progress in Theoretical Computer Science. Birkhäuser, 1993.

[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* 29^{th} *Symposium on Theory of Computing*, pages 11–20. ACM Press, 1997.

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

[Rei01] S. Reith. *Generalized Satisfiability Problems.* PhD thesis, University of Würzburg, 2001.

[RV00] S. Reith and H. Vollmer. Optimal satisfiability for propositional calculi and constraint satisfaction problems. In *Proceedings* 25^{th} *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, October 25–28, 1999.

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

[Tor00] J. Torán. On the hardness of graph isomorphism. In *Proceedings* 41^{st} *Foundations of Computer Science*, pages 180–186, 2000.

[Zwi98] U. Zwick. Finding almost-satisfying assignments. In *Proceedings* 30^{th} *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. Ehrenberg, 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.