Minimization for Generalized Boolean Formulas [*]

Edith Hemaspaandra
Department of Computer Science
Rochester Institute of Technology
Rochester, NY 14623, USA

Henning Schnoor
Institute for Computer Science
Christian-Albrechts-Universität zu Kiel
Kiel, Germany

Abstract

The minimization problem for propositional formulas is an important optimization problem in the second level of the polynomial hierarchy. In general, the problem is Σ_2^p-complete under Turing reductions, but restricted versions are tractable. We study the complexity of minimization for formulas in two established frameworks for restricted propositional logic: The Post framework allowing arbitrarily nested formulas over a set of Boolean connectors, and the constraint setting, allowing generalizations of CNF formulas. In the Post case, we obtain a dichotomy result: Minimization is solvable in polynomial time or coNP-hard. This result also applies to Boolean circuits. For CNF formulas, we obtain new minimization algorithms for a large class of formulas, and give strong evidence that we have covered all polynomial-time cases.

1 Introduction

The minimization problem for propositional formulas is one of the most natural optimization problems in the polynomial hierarchy. In fact, a variant of this problem was a major motivation for the definition of the polynomial hierarchy [Meyer and Stockmeyer, 1972]. The goal of minimization is to find a minimum equivalent formula to a given input formula. In this paper, we study the *minimum equivalent expression (MEE)* problem, where the input is a formula φ and a number k, and the question is to determine whether there exists a formula which is equivalent to φ and of size at most k (we study different notions of "size").

The problem is trivially in Σ_2^p, but a better lower bound than coNP-hardness had been open for many years. In [Hemaspaandra and Wechsung, 2002], the MEE problem was shown to be (many-one) hard for parallel access to NP. Recently, it was shown to be Σ_2^p-complete under Turing reductions in [Buchfuhrer and Umans, 2011].

Minimization in restricted fragments of propositional logic has been studied for the case of Horn formulas in order to find small representations of knowledge bases [Hammer and Kogan, 1995]. Prime implicates, a central tool for minimizing Boolean formulas [Quine, 1952], have been used in several areas of artificial intelligence research. We mention [Adjiman *et al.*, 2006], where prime implicates were used in peer-to-peer data management systems for the semantic web, and [Bittencourt, 2008], which applies them in the context of belief change operators. Two-level logic minimization is an important problem in logic synthesis [Umans *et al.*, 2006]. Different variants of minimization have been studied: The problem is Σ_2^p-complete for CNF formulas [Umans, 2001], NP-complete for Horn formulas [Boros and Čepek, 1994], and solvable in P for 2CNF formulas [Chang, 2004].

In this paper we study the complexity of minimization for syntactically restricted formulas. Two frameworks for restricting the expressive power of propositional logic have been used for complexity classifications in recent years:

- The *Post framework* [Post, 1941] considers formulas that instead of the usual operators $\wedge$, $\vee$, and $\neg$, use an arbitrary set B of Boolean functions as connectors. Depending on B, the resulting formulas may express only a subset of all Boolean functions, or may be able to express all functions more succinctly than the usual set $\{\wedge, \vee, \neg\}$.
- The *constraint framework* [Schaefer, 1978] studies formulas in CNF, where the types of allowed clauses (e.g., Horn, 3CNF, or XOR clauses) are defined in a *constraint language* Γ containing "templates" of generalized CNF-clauses that are allowed in so-called Γ-formulas.

In both frameworks, a wide range of complexity classifications has been obtained. For the Post framework, we mention the complexity of satisfiability [Lewis, 1979], equivalence [Reith, 2001], modal satisfiability [Hemaspaandra *et al.*, 2010], and non-monotonic logics [Thomas and Vollmer, 2010]. In the constraint setting, besides the satisfiability problem [Schaefer, 1978; Allender *et al.*, 2009], also enumeration of solutions [Creignou and Hébrard, 1997], equivalence and isomorphism [Böhler *et al.*, 2002], circumscription [Nordh and Jonsson, 2004], and many other problems have been studied, see [Creignou and Vollmer, 2008] for a survey. The complexity of satisfiability for non-Boolean domains is also a very active field, see e.g., [Bulatov, 2006; Bulatov and Valeriote, 2008].

For many considered problems, "dichotomy results" were

[*]Supported in part by NSF grants CCR-0311021 and IIS-0713061, the DAAD postdoc program, and by a Friedrich Wilhelm Bessel Research Award. Work done in part while Henning Schnoor was at the Rochester Institute of Technology.

achieved, proving that every choice of B or Γ leads to one of the same two complexity degrees, usually polynomial-time solvable and NP-complete. This is surprising since there are infinitely many sets B and Γ, and we know that there are, for example, infinitely many degrees of complexity between P and NP cases unless $\mathrm{P} = \mathrm{NP}$ [Ladner, 1975].

A "Galois Connection" between constraint languages and closure properties in the Post setting determines the complexity of many computational problems [Jeavons *et al.*, 1997; Schnoor and Schnoor, 2008]. In contrast, we show that these tools do not apply to minimization.

In the Post setting, we obtain a complete classification of the tractable cases of the minimization problem: For a set B of Boolean functions, the problem to minimize B-formulas is solvable in polynomial time or coNP-hard, hence avoiding the degrees between P and coNP-completeness. Our results in this framework apply to both the formula and the circuit case, and to different notions of size of formulas and circuits.

In the constraint case, we define *irreducible* constraint languages, among which we identify a large class whose formulas can be minimized in polynomial time, and prove NP- or coNP-hardness results for most of the remaining cases. More precisely, we prove the following: For an irreducible language for which equivalence can be tested efficiently, minimization is NP-complete if the language can express (dual) positive Horn, and can be solved in polynomial time otherwise. Our analysis thus implies that the NP-completeness result for positive Horn shown in [Boros and Čepek, 1994] is "optimal:" As soon as a CNF fragment of propositional logic is strictly less expressive than positive Horn, formulas can be minimized efficiently. Since irreducibility is a natural condition for constraint languages that are used in knowledge representation, a consequence of our result is that knowledge bases that do not need the full expressive power of positive Horn admit efficient "compression algorithms."

Our contribution is threefold:

1. We give new and non-trivial minimization algorithms for large classes of formulas.
2. In the Post setting, we prove that all remaining cases are coNP-hard. In the constraint setting, we give strong evidence that larger classes do not have efficient minimization algorithms.
3. We show that minimization behaves very differently than many other problems in the context of propositional formulas: The usually-applied algebraic tools for the constraint setting cannot be applied to minimization. Also, complexities in the Post- and constraint framework differ strongly: In particular, the constraint framework contains NP-complete cases; such cases do not exist in the Post framework (unless $\mathrm{NP} = \mathrm{coNP}$).

Due to space restrictions, most proofs can only be found in the technical report [Hemaspaandra and Schnoor, 2011].

2 Minimization in the Post Framework

We fix a finite set B of finitary Boolean functions. We define *B-formulas* inductively: A variable x is a B-formula, and if $\varphi_1, \dots, \varphi_n$ are B-formulas, and f is an n-ary function from B, then $f(\varphi_1, \dots, \varphi_n)$ is a B-formula. We often identify the function f and the symbol representing it. $\mathrm{VAR}(\varphi)$ denotes the set of variables in a formula φ. We write $\varphi(x_1, \dots, x_n)$ to indicate that $\mathrm{VAR}(\varphi) = \{x_1, \dots, x_n\}$. For an assignment $\alpha\colon \mathrm{VAR}(\varphi) \to \{0,1\}$, the *value of φ for* α, $\varphi(\alpha)$, is defined in the straightforward way. We write $\alpha \models \varphi$ if $\varphi(\alpha) = 1$, and say that α *satisfies* φ. Formulas φ_1 and φ_2 are *equivalent* if $\varphi_1(\alpha) = \varphi_2(\alpha)$ for all α, we then write $\varphi_1 \equiv \varphi_2$. The satisfiability problem for B-formulas is denoted with $\mathsf{SAT}(B)$.

Formulas can be succinctly represented as *circuits*, which are essentially DAGs where formulas are trees. Although every circuit can be rewritten into a formula, the size of the resulting formula can be exponential in the size of the circuit.

In the Post framework, we study two variations of the minimization problem that differ in the notion of the *size* of a formula φ. An obvious way to measure size is the number of occurrences of literals, which we denote with $size_l(\varphi)$. The second measurement is motivated by the study of Boolean circuits, where the size of a circuit is usually the number of non-input gates. For a formula, this is the number of appearing function symbols. We denote this number with $size_s(\varphi)$. Our results also hold for obvious variations of these measures (e.g., counting variables instead of occurrences, also counting input gates, etc). For a set B as above, we define:

Problem:	$\mathsf{MEE}^{F/C}_{l/s}(B)$
Input:	A B-formula/circuit ϕ and a number k
Question:	Is there a B-formula/circuit ψ with $size_{l/s}(\psi) \leq k$ and $\phi \equiv \psi$?

An n-ary Boolean function f is an OR-function if it is constant or if $f(x_1, \dots, x_n)$ is equivalent to $x_{r_1} \vee x_{r_2} \vee \dots \vee x_{r_m}$ for a subset $\{x_{r_1}, x_{r_2}, \dots, x_{r_m}\} \subseteq \{x_1, \dots, x_n\}$. AND- and XOR-functions are defined analogously. We show that formulas using only these functions can be minimized easily:

Theorem 2.1 *$\mathsf{MEE}^{F/C}_{l/s}(B)$ can be solved in polynomial time if B contains only OR-functions, only AND-functions, or only XOR-functions.*

We mention that the theorem, as all of our results in this section, applies to all four combinations of F/C and s/l. We also stress that all algorithms in this paper do not only determine whether a formula with the given size restriction exists, but also compute a minimum equivalent formula.

The satisfiability problem for the above cases can easily be solved in polynomial time. We now show that this is indeed a prerequisite for a tractable minimization problem—formally, we prove that the complement of the satisfiability problem (i.e., the set of all binary strings that are not positive instances of $\mathsf{SAT}(B)$) reduces to the minimization problem.

Theorem 2.2 *For every finite set B of Boolean functions, $\overline{\mathsf{SAT}(B)} \leq^{\log}_{m} \mathsf{MEE}^{F/C}_{l/s}(B)$.*

Using results on the complexity of $\mathsf{SAT}(B)$ [Lewis, 1979], we obtain hardness results for a large class of sets B:

Corollary 2.3 *Let B be a finite set of Boolean functions such that there is a B-formula that is equivalent to $x \wedge \overline{y}$. Then $\mathsf{MEE}^{F/C}_{l/s}(B)$ is coNP-hard.*

The remaining cases are those where satisfiability is tractable, but which are not of the forms covered by Theorem 2.1. We show that in these cases, minimization is coNP-hard using a reduction from the *equivalence problem* for formulas, which asks to determine whether two given formulas are equivalent. The proof of the theorem below relies on the following idea: Given two formulas as input for the equivalence problem, we combine them into a single formula which is "trivial" if the formulas are equivalent, but "complicated" otherwise. The "gap" between the cases is large enough to yield a reduction to the minimization problem.

Theorem 2.4 $\mathsf{MEE}^{F/C}_{l/s}(B)$ *is* coNP*-hard if one of the following is true:*

- *There is a B-formula equivalent to $x \wedge y$, and a $B \cup \{1\}$-formula equivalent to $x \vee y$,*
- *there is a B-formula equivalent to $(x \wedge y) \vee (x \wedge z) \vee (y \wedge z)$.*

Proof Sketch. We only handle the first case. Testing whether two B-formulas H_1 and H_2 are equivalent is coNP-hard ([Reith, 2001]). We reduce this problem to $\mathsf{MEE}^{F/C}_{s}(B)$. By assumption, there are B-formulas $f_\vee(x, y, t)$ and $f_\wedge(x, y)$ with $f_\wedge(x, y) \equiv x \wedge y$, and $f_\vee(x, y, 1) \equiv x \vee y$. Let m be the maximal arity of a function in B. For a new variable t, define

- $l = size_s(f_\wedge(H_1, t))$, we assume $l > 1$.
- Z is equivalent to $\bigwedge_{i=1}^{m \cdot l} z_i$ for new variables z_i.
- $G = f_\wedge(f_\vee(f_\wedge(H_1, H_2), f_\wedge(f_\vee(H_1, H_2, t), Z), t), t)$.

G can be computed in polynomial time using a logarithmic-depth tree construction. One can show that $H_1 \equiv H_2$ if and only if $\langle G, l\rangle \in \mathsf{MEE}^{F/C}_{s}(B)$. The proof for $\mathsf{MEE}^{F/C}_{l}(B)$ uses a similar construction. □

Using algebraic techniques, the above implies that if B contains functions f, g, and h such that f that is not an OR-function, g is not an AND-function, and h is not an XOR-function, then minimization is coNP-hard. This applies to all cases except those covered by our polynomial-time results. We therefore obtain the following full classification:

Corollary 2.5 *Let B be a finite set of Boolean functions.*

- *If B contains only OR-functions, only AND-functions, or only XOR-functions, then $\mathsf{MEE}^{F/C}_{l/s}(B)$ can be solved in polynomial time.*
- *Otherwise,* $\mathsf{MEE}^{F/C}_{l/s}(B)$ *is* coNP*-hard.*

3 Minimization in the CNF framework

Constraint formulas are CNF-formulas, where the set of allowed types of clauses is defined in a *constraint language* Γ, which is a finite set of non-empty finitary Boolean relations. A Γ-clause is of the form $R(x_1, \dots, x_n)$, where R is an n-ary relation from Γ, and $x_1, \dots, x_n$ are variables. A Γ*-formula* is a conjunction of Γ-clauses, it is satisfied by α, if for every clause $R(x_1, \dots, x_n)$ in φ, we have that $(\alpha(x_1), \dots, \alpha(x_n)) \in R$. A relation R is *expressed* by a formula if the tuples in the relation are exactly the solutions of the formula (with some canonical order on the variables). We denote the satisfiability problem for Γ-formulas with $\mathsf{SAT}(\Gamma)$.

A natural way to measure the size of a CNF formula is the number of clauses—for a fixed language Γ, this is linearly related to the number of variable occurrences. We thus consider the following problem:

Problem:	$\mathsf{MEE}\,(\Gamma)$
Input:	A Γ-formula φ, an integer k
Question:	Is there a Γ-formula ψ with at most k clauses and $\psi \equiv \varphi$?

To state our classification, we recall relevant properties of Boolean relations (for more background on these properties and how they relate to complexity classifications of constraint-related problems, see e.g., [Creignou *et al.*, 2001]).

1. A relation is *affine* if it can be expressed by a $\{x, \overline{x}, x_1 \oplus \cdots \oplus x_n, \neg(x_1 \oplus \cdots \oplus x_n) \mid n \in \mathbb{N}\}$-formula.
2. A relation is *bijunctive* if it can be expressed by a Γ^2-formula, where Γ^2 is the set of binary Boolean relations.
3. A relation is *Horn* if it can be expressed by a $\{x, \overline{x}, (x_1 \wedge \cdots \wedge x_n \rightarrow y), (\overline{x_1 \wedge \cdots \wedge x_n}) \mid n \in \mathbb{N}\}$-formula.
4. A relation is *positive Horn* if it can be expressed by a $\{x_1 \wedge \cdots \wedge x_n \rightarrow y \mid n \in \mathbb{N}\}$-formula.
5. A relation is *IHSB+* if it can be expressed by a $\{x, \overline{x}, x \rightarrow y, (x_1 \vee \cdots \vee x_n) \mid n \in \mathbb{N}\}$-formula.

A constraint language Γ is affine, bijunctive, etc., if every relation in Γ is. Γ is *dual (positive) Horn* if $\overline{\Gamma}$ is (positive) Horn, and *IHSB−* if $\overline{\Gamma}$ is IHSB+. Here $\overline{\Gamma}$ is the *dual* of Γ, obtained from Γ by swapping 0 and 1 in all relations in Γ. Additionally, Γ is *Schaefer* if it is affine, bijunctive, Horn, or dual Horn. This property implies tractability of many problems for Boolean constraint languages, including satisfiability [Schaefer, 1978], equivalence [Böhler *et al.*, 2002] and enumeration [Creignou and Hébrard, 1997].

3.1 Irreducible Relations

In many cases, if constraint languages Γ_1 and Γ_2 "have the same expressive power," then problems for Γ_1 and Γ_2 have the same complexity. We show that this is not true for minimization, even for a very strict version of "having the same expressive power" (see [Schnoor and Schnoor, 2008] for details on notions of expressive power).

Example 3.1 Let $\Gamma_1 := \{x \vee y\}$ and $\Gamma_2 := \{(x \vee y), (x \vee (y \wedge z)), (x \vee (y \wedge z \wedge w))\}$.

- Γ_1 and Γ_2 have the same expressive power, since $y \vee (x_1 \wedge \cdots \wedge x_n) \equiv (y \vee x_1) \wedge \cdots \wedge (y \vee x_n)$.
- $\mathsf{MEE}\,(\Gamma_1)$ can be solved in polynomial time.
- $\mathsf{MEE}\,(\Gamma_2)$ is NP-hard by a reduction from Vertex Cover for cubic graphs: A cubic graph $G = (V, E)$ has a vertex cover of size k if and only if the formula $\bigwedge_{\{i,j\} \in E}(x_i \vee x_j)$ has an equivalent Γ_2-formula with k clauses.

Hence unlike most problems in the constraint context, the complexity of minimization is not determined by the expressive power of a constraint language. This is the reason why our analysis cannot simply follow the well-established line of reasoning used to obtain complexity classifications in the literature. Considering the above example, the problems in minimizing Γ_2-formulas are combinatorial and do not stem from the difficulty of determining a "minimum representation" of a given formula. Therefore the NP-hardness is not in finding a minimum representation of the formula, but from the difficulty to use the available "building blocks" efficiently.

In the example above, the problems arise because Γ_2 contains "combined" relations which can be rewritten into simpler clauses: $(x \vee (y \wedge z))$ is equivalent to $(x \vee y) \wedge (x \vee z)$. An important feature of constraint formulas is that they build formulas from "local conditions" expressed in the individual clauses. The clause $(x \vee (y \wedge z))$ is in a way not "as local as it can be," since it can be rewritten as the conjunction of two "easier" conditions. We define *irreducible* relations as those that cannot be rewritten in this way:

Definition An n-ary relation R is *irreducible*, if for every formula $R_1(x_1^1 \dots, x_{k_1}^1) \wedge \dots \wedge R_m(x_1^m, \dots, x_{k_m}^m)$ (where each R_i is a k_i-ary Boolean relation) which is equivalent to $R(x_1, \dots, x_n)$, one of the R_i-clauses has arity at least n. A constraint language Γ is *irreducible* if every relation in Γ is.

Irreducibility is a rather natural condition—in fact, most relations usually considered in the constraint context meet this definition. Irreducible languages only allow "atomic" clauses that cannot be split up further. In practice, for example in the design of knowledge bases, irreducible languages are more likely to be used: They provide users with atomic constructs as a basis from which more complex expressions can be built.

3.2 Polynomial-Time Cases

We prove polynomial-time results for every case in which such a result is conceivable: The MEE problem for positive Horn formulas is NP-complete [Boros and Čepek, 1994]. We show that for every irreducible constraint language that is Schaefer, and does not have all the expressive power of positive Horn (or dual positive Horn), the minimization problem can be solved efficiently. For non-Schaefer languages, even testing equivalence is coNP-hard, and hence an efficient minimization procedure cannot be expected. Following known structural results about Boolean constraint languages, there are three cases to consider: The case where Γ is affine, bijunctive, or IHSB+. We provide polynomial-time algorithms for all of these cases.

IHSB+ and IHSB− formulas

We start our polynomial-time results with the most involved of these constructions, proving that irreducible constraint languages that are IHSB+ lead to an easy minimization problem (the IHSB− case is analogous). Requiring irreducibility is necessary: The language Γ_2 discussed in Example 3.1 is IHSB+ (even considerably less expressive than IHSB+), but, as argued before, has an NP-hard minimization problem.

The main idea of the algorithm is the following: We rewrite formulas using multi-ary OR, implication, equality, and literals into conjunctions of, to a large degree, independent formulas, each containing only OR, implications, equalities, or literals. Each of these formulas then can be minimized locally with relatively easy algorithms. The main task that our algorithm performs is "separating" the components of the input formula in such a way that minimizing the mentioned subformulas locally is equivalent to minimizing the entire formula.

Theorem 3.2 *Let* $\Gamma = \{\rightarrow, =, x, \overline{x}\} \cup \{\mathsf{OR}^m \mid m \leq k\}$ *for some* $k \in \mathbb{N}$. *Then* $\mathsf{MEE}(\Gamma) \in \mathrm{P}$.

Proof. For variables u and v, we write $u \rightsquigarrow_\varphi v$ (u *leads to* v *in* φ) if there is a directed $\{\rightarrow, =\}$-path in φ from u to v, we often simply write $u \rightsquigarrow v$. Similarly, if there are OR-clauses $C_1 = (x_1 \vee \dots \vee x_n)$ and $C_2 = (y_1 \vee \dots \vee y_m)$, we write $C_1 \rightsquigarrow C_2$ if every of the x_i leads to one of the y_j. Since satisfiability for Γ-formulas can be tested in polynomial time, we assume that all occurring formulas are satisfiable. For any Γ-formula φ, let φ_{OR} denote the formula obtained from φ by removing every clause that is not an OR-clause with at least 2 variables, and let $\varphi_\rightarrow$ be the conjunction of all implication-clauses in φ, φ_{lit} the literals in φ, and $\varphi_=$ the equality clauses.

We now describe the minimization procedure. We use some canonical way of ordering variables and clauses and repeat the following steps until no changes occur anymore:

```
1:  **Input**: $\Gamma$-formula $\varphi$
2:  **while** changes still occur **do**
3:      For a set of variables connected with $=$, only keep the minimal variable in non-equality clauses (by variable identification)
4:      **if** there exist OR-clauses $C_1 \neq C_2$ with $C_1 \rightsquigarrow C_2$, **then**
5:          If $C_2 \rightsquigarrow C_1$, then remove the minimal of the two
6:          Otherwise, remove $C_2$
7:      **end if**
8:      **if** there is clause $(x_1 \vee \dots \vee x_n)$, variable $v$ with $x_i \rightsquigarrow v$ for all $i$, **then**
9:          introduce clause $v$
10:         remove $\rightarrow$-clauses leading to $v$
11:     **end if**
12:     **if** literal $x$ occurs, $x \rightsquigarrow y$ **then**
13:         replace final clause in path with $y$
14:     **end if**
15:     **if** literal $\overline{y}$ occurs, $x \rightsquigarrow y$ **then**
16:         replace first clause in path with $\overline{x}$
17:     **end if**
18:     Remove variables occurring as negative literals from OR-clauses
19:     **if** $(x_1 \vee \dots \vee x_n)$ is clause, $x_i \rightsquigarrow x_j$ for $i \neq j$ **then**
20:         remove $x_i$ from the clause
21:     **end if**
22:     **if** there are variables such that $x_1 \rightsquigarrow x_2, \dots, x_{n-1} \rightsquigarrow x_n, x_n \rightsquigarrow x_1$ **then**
23:         exchange implications between them with equalities.
24:     **end if**
25:     **if** $u$ ($\overline{u}$) appears as a literal **then**
26:         remove clauses of the form $(v \rightarrow u)$ (($u \rightarrow v$)).
27:     **end if**
28:     Locally minimize $\varphi_=$ and $\varphi_\rightarrow$.
29: **end while**
```

Note that $\varphi_=$ and φ_{lit} can be minimized trivially, and $\varphi_\rightarrow$ can be minimized due to a result from [Aho *et al.*, 1972], since finding a transitive reduction of a directed graph is exactly the problem of minimizing a formula in which only implications of positive literals appear. It can be shown that the algorithm produces a formula that is equivalent to the input and minimal with respect to the number of clauses. □

A careful analysis of the proof yields that it also holds true if Γ does not contain all the relations defining IHSB+, even though in these cases, only a restricted vocabulary is available for the minimum formula, and also applies in the cases where the relations are not the ones mentioned in the theorem, but are still IHSB+ (the IHSB− case is analogous):

Corollary 3.3 *Let Γ be an irreducible constraint language which is IHSB+ or IHSB−. Then* $\mathsf{MEE}(\Gamma) \in \mathrm{P}$.

We mention that the above cases remain polynomial-time if the input formula may use clauses of unbounded arity.

Bijunctive Formulas

In a similar way, we show that irreducible bijunctive constraint languages give a tractable minimization problem:

Theorem 3.4 *Let Γ be a constraint language which is irreducible and bijunctive. Then* $\mathsf{MEE}(\Gamma) \in \mathrm{P}$.

Affine Formulas

We conclude our polynomial-time results with the affine case. Affine formulas represent linear equations over $\mathrm{GF}(2)$. We therefore can apply standard linear algebra techniques to obtain an efficient minimization algorithm.

Theorem 3.5 *Let Γ be an irreducible and affine constraint language. Then* $\mathsf{MEE}(\Gamma) \in \mathrm{P}$.

3.3 Lower Bounds

As mentioned before, our polynomial-time results cover all cases where polynomial-time algorithms can be expected. We now prove hardness results for most of the remaining cases.

Minimization and Satisfiability

If a constraint language Γ is not Schaefer (i.e., neither Horn, dual Horn, bijunctive, nor affine), then the satisfiability problem for $\Gamma^+ = \Gamma \cup \{x, \overline{x}\}$ (Γ extended with literals) is NP-complete. Since the natural analog of Theorem 2.2 can also be shown in the CNF setting, we obtain the following:

Corollary 3.6 *Let Γ be a constraint language that is not Schaefer. Then* $\mathsf{MEE}(\Gamma^+)$ *is* coNP*-hard.*

NP-completeness Results

We use results on the hardness of minimizing Horn formulas to obtain an NP-completeness result for a large class of constraint languages: An irreducible constraint language that is not covered by our polynomial-time results, but that is Schaefer, leads to an NP-complete minimization problem. The NP upper bound is clear, since the equivalence problem can be solved in polynomial time in these cases. The proof of the hardness result is more involved, and first establishes a characterization of the relations contained in the covered constraint languages, which shows that in all cases, we have a relation that is expressive enough to encode minimization for positive Horn formulas. We thus obtain the following result:

Theorem 3.7 *Let Γ be an irreducible constraint language that is Schaefer, not affine, not bijunctive, not IHSB+, and not IHSB−. Then* $\mathsf{MEE}(\Gamma)$ *is* NP*-complete.*

3.4 Classification Theorem

The analysis in the previous sections yields the following classification:

Theorem 3.8 *Let Γ be an irreducible constraint language.*

1. *If Γ is affine, bijunctive, IHSB+, or IHSB−, then* $\mathsf{MEE}(\Gamma) \in \mathrm{P}$.
2. *Otherwise, if Γ is Horn or dual Horn, then* $\mathsf{MEE}(\Gamma)$ *is* NP*-complete,*
3. *Otherwise, Γ is not Schaefer, and* $\mathsf{MEE}(\Gamma^+)$ *is* coNP*-hard.*

While the theorem does not completely classify the complexity of the MEE problem for all irreducible constraint languages, we consider it unlikely that there exist more polynomial-time cases than the ones we discovered: To the best of our knowledge, no decision problem for non-Schaefer languages has been proven to be in polynomial time except for trivial cases (e.g., satisfiability is trivial if all relevant relations contain the all-0 tuple). Also, for non-Schaefer languages Γ, already testing equivalence of formulas is coNP-hard. This implies that, unless $\mathrm{P} = \mathrm{NP}$, there cannot be a polynomial-time algorithm that, given a Γ-formula, computes its "canonical" (i.e., up to differences checkable by a polynomial-time algorithm) minimum equivalent expression. We are therefore confident that our classification covers all polynomial-time cases for irreducible constraint languages.

It is worth noting that the prerequisite that Γ is irreducible is certainly required for the polynomial-time cases, as the earlier example highlighted. For the hardness results, this is less clear—the coNP-hardness does not rely on this prerequisite at all, and for the NP-complete Horn cases, we consider it unlikely that there is a constraint language with the same expressive power that does not directly encode positive Horn.

4 Conclusion and Open Questions

We studied the complexity of minimization for restricted classes of propositional formulas in two settings, obtained a complete characterization of all tractable cases in the Post case, and a large class of tractable cases in the constraint case.

Open questions include the exact classification of the coNP-hard cases. It is likely that most of them are NP-hard as well. It would be very interesting to determine whether some of these are actually Σ_2^p-complete (this does not follow directly from the Σ_2^p-completeness of the minimization problem for CNF formulas [Umans, 2001], since our constraint languages Γ and bases B are finite). Finally, non-irreducible constraint languages are an interesting open issue.

Acknowledgment

We thank the anonymous reviewers for many helpful comments, in particular for pointing out an issue with the definition of irreducibility.

References

[Adjiman *et al.*, 2006] P. Adjiman, P. Chatalic, F. Goasdoué, M.-C. Rousset, and L. Simon. Distributed reasoning in a peer-to-peer setting: Application to the semantic web. *JAIR*, 25:269–314, 2006.

[Aho *et al.*, 1972] A. Aho, M. Garey, and J. Ullman. The transitive reduction of a directed graph. *SIAM Journal on Computing*, 2(1):131–137, 1972.

[Allender *et al.*, 2009] E. Allender, M. Bauland, N. Immerman, H. Schnoor, and H. Vollmer. The complexity of satisfiability problems: Refining Schaefer's Theorem. *Journal of Computer and System Sciences*, 75(4):245–254, 2009.

[Bittencourt, 2008] G. Bittencourt. Combining syntax and semantics through prime form representation. *J. Log. Comput.*, 18(1):13–33, 2008.

[Böhler *et al.*, 2002] E. Böhler, E. Hemaspaandra, S. Reith, and H. Vollmer. Equivalence and isomorphism for Boolean constraint satisfaction. In *Proc. CSL*, volume 2471 of *LNCS*, pages 412–426. Springer Verlag, 2002.

[Boros and Čepek, 1994] E. Boros and O. Čepek. On the complexity of Horn minimization. Technical Report 1-94, RUTCOR Research Report RRR, Rutgers University, New Brunswick, NJ, January 1994.

[Buchfuhrer and Umans, 2011] D. Buchfuhrer and C. Umans. The complexity of boolean formula minimization. *Journal of Computer and Systems Sciences*, 77(1):142–153, 2011.

[Bulatov and Valeriote, 2008] A. Bulatov and M. Valeriote. Recent results on the algebraic approach to the csp. In N. Creignou, P. Kolaitis, and H. Vollmer, editors, *Complexity of Constraints*, volume 5250 of *LNCS*, pages 68–92. Springer, 2008.

[Bulatov, 2006] A. Bulatov. A dichotomy theorem for constraint satisfaction problems on a 3-element set. *J. ACM*, 53(1):66–120, 2006.

[Chang, 2004] T. Chang. Horn formula minimization. Master's thesis, Rochester Institute of Technology, 2004.

[Creignou and Hébrard, 1997] 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.

[Creignou and Vollmer, 2008] N. Creignou and H. Vollmer. Boolean constraint satisfaction problems: When does Post's lattice help? In N. Creignou, P. Kolaitis, and H. Vollmer, editors, *Complexity of Constraints*, pages 3–37. Springer Verlag, Berlin Heidelberg, 2008.

[Creignou *et al.*, 2001] N. Creignou, S. Khanna, and M. Sudan. *Complexity Classifications of Boolean Constraint Satisfaction Problems*. Monographs on Discrete Applied Mathematics. SIAM, 2001.

[Hammer and Kogan, 1995] P. Hammer and A. Kogan. Quasi-acyclic propositional Horn knowledge bases: Optimal compression. *IEEE Trans. Knowl. Data Eng.*, 7(5):751–762, 1995.

[Hemaspaandra and Schnoor, 2011] E. Hemaspaandra and H. Schnoor. Minimization for generalized Boolean formulas. Technical Report arXiv:1104.2312, Computing Research Repository, 2011.

[Hemaspaandra and Wechsung, 2002] E. Hemaspaandra and G. Wechsung. The minimization problem for Boolean formulas. *SIAM J. Comput.*, 31(6):1948–1958, 2002.

[Hemaspaandra *et al.*, 2010] E. Hemaspaandra, H. Schnoor, and I. Schnoor. Generalized modal satisfiability. *Journal of Computer and System Sciences*, 76(7):561—578, 2010.

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

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

[Lewis, 1979] H. Lewis. Satisfiability problems for propositional calculi. *Math. Systems Theory*, 13:45–53, 1979.

[Meyer and Stockmeyer, 1972] A. Meyer and L. Stockmeyer. The equivalence problem for regular expressions with squaring requires exponential time. In *Proceedings 13th Symposium on Switching and Automata Theory*, pages 125–129. IEEE Computer Society Press, 1972.

[Nordh and Jonsson, 2004] G. Nordh and P. Jonsson. An algebraic approach to the complexity of propositional circumscription. In *Proc. LICS*, pages 367–376, 2004.

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

[Quine, 1952] W. V. Quine. The problem of simplifying truth functions. *The American Mathematical Monthly*, 59(8):521–531, 1952.

[Reith, 2001] S. Reith. *Generalized Satisfiability Problems*. PhD thesis, Fachbereich Mathematik und Informatik, Universität Würzburg, 2001.

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

[Schnoor and Schnoor, 2008] H. Schnoor and I. Schnoor. Partial polymorphisms and constraint satisfaction problems. In N. Creignou, P. Kolaitis, and H. Vollmer, editors, *Complexity of Constraints*, volume 5250 of *LNCS*, pages 229–254. Springer, 2008.

[Thomas and Vollmer, 2010] M. Thomas and H. Vollmer. Complexity of non-monotonic logics. Technical Report arXiv:1009.1990, Computing Research Repository, 2010.

[Umans *et al.*, 2006] C. Umans, T. Villa, and A. Sangiovanni-Vincentelli. Complexity of two-level logic minimization. *IEEE Trans. Computer-Aided Design of Int. Circuits and Systems*, 25(1):1230–1246, 2006.

[Umans, 2001] C. Umans. The minimum equivalent DNF problem and shortest implicants. *Journal of Computer and Systems Sciences*, 63(4):597–611, 2001.