Inverse Problems Have Inverse Complexity

Tobias Berg and Harald Hempel

Fakultät für Mathematik und Informatik
Friedrich-Schiller-Universität Jena
07740 Jena, Germany
tberg@minet.uni-jena.de
hempel@uni-jena.de

Abstract. In this paper we show that inverting problems of higher complexity is easier than inverting problems of lower complexity. While inverting Σ_i^p3CNFSAT is known to be coNP-complete [6] for $i = 1$ we prove that it remains coNP-complete for $i = 2$ and is in P for all $i \geq 3$. Relatedly, we show that inverting Σ_i^p3DNFSAT is in P for all $i \geq 1$.

1 Introduction

Do problems of higher complexity also always have inverse problems of higher complexity? We answer this question to the negative by showing that within the polynomial hierarchy complete problems from higher levels have easier inverse problems than those from lower levels. More precisely, we prove that while inverting Σ_i^p3CNFSAT is coNP-complete for $i = 1$ [6] it is also coNP-complete for $i = 2$, yet is in P for all $i \geq 3$. In contrast, inverting Σ_i^p3DNFSAT is easy, i.e., in P, for all $i \geq 1$.

Standard NP decision problems A are of the nature given an object x find out if there exists a proof for the membership of x in A. The inverse problem would then be given a set of proofs for membership in A does there exist an object x such the proofs for membership of x in A are exactly the given ones? For example, while the well known satisfiability problem SAT asks if a given Boolean formula has a satisfying assignment, the computational problem INVERSE SAT is defined as follows: Given a set of assignments does there exist a Boolean formula F such that the given assignments are exactly the satisfying assignments of F. While INVERSE SAT is (trivially) in P it has been shown that INVERSE 3SAT is coNP-complete [6]. Note that our proofs showing that inverting Σ_i^p3CNFSAT as well as inverting Σ_j^p3DNFSAT is in P for $i \geq 3$ and $j \geq 1$, respectively, are constructive. Hence, not only the decision if a formula F such that the given assignments are exactly the satisfying assignments of F exists but also actually finding F can be done in polynomial time.

In general, the study of inverse problems contributes to the field of identifying meaningful structures in data and efficient knowledge representation. Finding a computationally appealing representation for a given set of data can only be an easy problem if the corresponding inverse problem is easy. Furthermore, the study of inverse NP-problems may be helpful in gaining more insight into the nature of NP-completeness and may also be helpful in characterizing "natural" verifiers.

It has been shown that for many NP-complete problems inverting their natural verifier is coNP-complete [6, 2, 7]. However, the complexity of inverse problems in general heavily depends on the underlying verifier [2]. Formally, NP is the set of all languages A such that there exists a polynomial time computable 2-ary predicate V (also called a polynomial-time-verifier or NP-verifier) such that for all $x \in \Sigma^*$ we have $x \in A$ if and only if there exists a polynomial size bounded string π such that $(x, \pi) \in V$. The inverse NP-problem of A relative to V, INVS_V, is given a set of strings $\{\pi_1, \pi_2, \ldots, \pi_k\}$ does there exist a string x such that $\{\pi_1, \pi_2, \ldots, \pi_k\} = \{\pi : (x, \pi) \in V\}$? There are NP-complete problems that have NP-verifiers that can be inverted in P while the inversion of other of their NP-verifiers is Σ_2^p-complete [2]. Despite these results we feel that studying the inverse problems relative to the canonical (natural) NP-verifiers will give the true answer concerning the complexity of the inverse problems.

In this paper we study inverse problems from the classes Σ_i^p from the polynomial hierarchy thereby giving answers to some open questions posed in [2]. We introduce the notion of a verifier for the classes Σ_i^p and define the inverse problem for such verifiers. After giving upper bounds for the complexity of these inverse problems based on Σ_i^p-verifiers we study the inverse problem for some specific Σ_i^p-complete satisfiability problems such as Σ_i^p3DNFSAT and Σ_i^p3CNFSAT, yielding the above mentioned results.

We mention in passing that inverse NP-problems in a slightly different setting, namely in a setting where the solutions are not given explicitely as a list but implicitely in form of a boolean circuit accepting exactly those solutions have been studied in [4]. Also, lower and upper bounds for the inversion of RE problems have been found by the authors [1].

This paper is organized as follows: After formally introducing some notation and giving some remarks on previous results in Section 2, we will translate these concepts to problems from Σ_i^p in Section 3. In Section 3 we will also give an upper bound for inverting a reasonable restricted subset of verifiers for Σ_i^p-languages. We will furthermore examine the inverse complexity of some natural verifiers for specific Σ_i^p-complete satisfiability problems yielding the above mentioned results that are interesting beyond the scope of inverse problems.

2 Preliminaries

We assume the reader to be familiar with the basic concepts and notations of complexity theory (see [9, 5]).

Our alphabet will be $\Sigma = \{0, 1\}$. For a string $\alpha \in \Sigma^*$ let α^i denote the ith letter of α, i.e., $\alpha = \alpha^1\alpha^2\alpha^3 \ldots \alpha^{|\alpha|}$. As is standard in complexity theory an assignment for a Boolean formula F with variables $x_1, x_2, \ldots, x_n$ is a length n

string $\alpha = \alpha^1\alpha^2\alpha^3\dots\alpha^n$ which, for all $1 \leq i \leq n$, assigns the Boolean value α^i to the variable x_i. Recall that a 3CNF (3DNF) formula is a Boolean formula in conjunctive (disjunctive) normal form having exactly 3 literals per clause.

Verifiers, the language associated with a verifier, sets of proofs, and inverse problems relative to a given verifier can in general be defined as follows:

Definition 1. 1. A relation V is called a verifier if and only if $V \subseteq \Sigma^* \times \Sigma^*$.
2. For any verifier V and any string $x \in \Sigma^*$, the set of proofs for x with respect to V, short $V(x)$, is defined as

$$V(x) = \{\pi \in \Sigma^* : (x, \pi) \in V\}.$$

3. The language associated with V, $L(V)$, is defined as

$$L(V) = \{x \in \Sigma^* : V(x) \neq \emptyset\}.$$

4. The inverse problem relative to a verifier V, INVS_V, is defined as

$$\text{INVS}_V = \{\Pi \subseteq \Sigma^* : (\exists x \in L(V))[V(x) = \Pi]\}.$$

One could also define the inverse problem as $\text{INVS}_V = \{\Pi \subseteq \Sigma^* : (\exists x \in \Sigma^*)[V(x) = \Pi]\}$. However, this marginal change in definition – adding the empty set to the inverse problem – should not result in any differences regarding the complexity of both types of inverse problems. We take the freedom to sometimes write $V(x, \pi)$ instead of $(x, \pi) \in V$ for verifiers V and strings x and π.

The class NP can be viewed as the class of languages having polynomial-time verifiers.

Definition 2. A verifier V is called a polynomial-time verifier if and only if

1. $V \in \text{P}$ and
2. there is a polynomial p such that for all $x, \pi \in \Sigma^*$, $(x, \pi) \in V \rightarrow |\pi| \leq p(|x|)$.

In this paper polynomial-time verifiers will also be called NP-verifiers. It is well-known that a language A is in NP if and only if there exists an NP-verifier V such that $A = L(V)$.

Inverse NP-problems are exactly the inverse problems relative to NP-verifiers and have been introduced in [2]. Clearly, it does not make sense to speak of inverting NP-problems without specifying the verifier. And in fact, inverting different NP-verifiers for one and the same NP-problem has different complexity. In [2] it has been shown that for every problem $A \in \text{NP}$ there exists an NP-verifier V such that $L(V) = A$ and $\text{INVS}_V \in P$. Here one can even show that there exists such a verifier that is fair [2].

Definition 3. [2] An NP-verifier V is called fair if and only if there exists a polynomial q such that for all $x \in L(V)$ there exists a string $x' \in L(V)$ such that $V(x) = V(x')$ and $|x'| \leq q(||V(x)||)$, where $||V(x)||$ denotes the length of the encoding of the set $V(x)$.

In contrast, it has been shown that several NP-problems have NP-verifiers such that the inverse problem relative to those verifiers is coNP-complete [6, 2, 7]. And it is also known that there is a tight Σ_2^p upper bound for inverting fair NP-verifiers [2], where Σ_2^p denotes the second level of the polynomial hierarchy.

Recall that for a complexity class $\mathcal{C}$ the classes $\mathrm{P}^{\mathcal{C}}$ and $\mathrm{NP}^{\mathcal{C}}$ are defined as the classes of languages that can be accepted by polynomial-time deterministic and nondeterministic, respectively, oracle Turing machines that make queries to a language from $\mathcal{C}$. Based on this concept the Σ_i^p levels of the polynomial-time hierarchy are defined as follows.

Definition 4. [8, 10] The complexity classes Σ_i^p are inductively defined via

1. $\Sigma_0^p = \mathrm{P}$ and
2. $\Sigma_{i+1}^p = \mathrm{NP}^{\Sigma_i^p}$ for all $i \geq 1$.

A useful characterization of the classes Σ_i^p was proven in [8].

Theorem 1. [8] *A language $A \subseteq \Sigma^*$ belongs to Σ_i^p if and only if there exists a predicate $V \in P$ and polynomials $p_1, ..., p_i$ such that for all $x \in \Sigma^*$ the following holds:*

$$x \in A \leftrightarrow (\exists y_1 \in \Sigma^*)(\forall y_2 \in \Sigma^*)(\exists y_3 \in \Sigma^*) \ldots (Qy_i \in \Sigma^*)[|y_1| \leq p_1(|x|) \wedge ... \wedge |y_i| \leq p_i(|x|) \wedge (x, y_1, ..., y_i) \in V].$$

If i is even then $Q = \forall$ and if i is odd then $Q = \exists$.

As we have pointed out earlier the complexity of inverse problems heavily depends on the underlying verifier. In order to study inverse NP-problems researchers have focused on inverting "natural" NP-verifiers, i.e., NP-verifiers that have proofs that closely reflect the canonic statement of the original NP-problem. For instance, in the case of SATISFIABILITY the most natural proof would be an assignment and a natural NP-verifier for SATISFIABILITY would be

$$V_{\mathrm{SAT}} = \{(F, \alpha) : F \text{ is a Boolean formula and } \alpha \text{ satisfies } F\}.$$

The first "natural" NP-verifiers have been studied in [6], where the complexity of inverting various syntactically constrained satisfiability problems has been studied. Following this line of research the coNP-completeness of the inverse problem (with respect to some "natural" NP-verifier) for some more NP-complete problems has been shown :

– 3SAT [6]
– CLIQUE, EXACT COVER, VERTEX COVER, SUBSET SUM (=KNAPSACK), STEINER TREE IN GRAPHS, PARTITION [2]
– HAMILTONIAN CIRCUIT, 3-D MATCHING [7]

For formal definition of these problems see [3].

3 The Inverse Problem for Σ_i^p

It has been suggested in [2] to examine the inverse problems for classes different than NP. In this section we will lay the ground for studying inverse Σ_i^p problems.

The class Σ_i^p is defined as the class of all languages that can be decided by a nondeterministic polynomial-time oracle Turing machine with queries to a Σ_{i-1}^p oracle, $\Sigma_i^p = \mathrm{NP}^{\Sigma_{i-1}^p}$. This leads to the following definition of a Σ_i^p-verifier.

Definition 5. A verifier V is called a Σ_i^p-verifier if and only if

1. $V \in \mathrm{P}^{\Sigma_{i-1}^p}$,
2. there exists a polynomial p such that for all $x, \pi \in \Sigma^*$, $(x, \pi) \in V \to |\pi| \leq p(|x|)$.

Observation 1 *For every language $A \subseteq \Sigma^*$, A is in Σ_i^p if and only if there exists a Σ_i^p-verifier V such that $L(V) = A$.*

Fair Σ_i^p-verifiers can be defined in analogy to Definition 3.

Definition 6. A Σ_i^p-verifier V is called a fair Σ_i^p-verifier if and only if there exists a polynomial q such that $(\forall x \in L(V))(\exists x' \in L(V))[V(x) = V(x') \wedge |x'| \leq q(||V(x)||)]$, where $||V(x)||$ denotes the length of the encoding of the set $V(x)$.

Informally, a Σ_i^p-verifier is called fair if for any set of proofs Π either

– there exists a polynomially length-bounded string (theorem) x' with exactly the proofs from Π or
– there exists no theorem with the set of proofs Π.

With this definitions in mind, what is an upper complexity bound for inverting a fair Σ_i^p-verifier?

Theorem 2. *If V is a fair Σ_i^p-verifier ($i \geq 1$), then* $\mathrm{INVS}_V \in \Sigma_{i+1}^p$.

Proof. The case $i = 1$ has been shown in [2]. So let $i \geq 2$ and let V be a fair Σ_i^p-verifier, i.e., $V \in \mathrm{P}^{\Sigma_{i-1}^p}$ and there exist two polynomials p and q such that

1. for all $x, \pi \in \Sigma^*$, $(x, \pi) \in V \to |\pi| \leq p(|x|)$
2. for all $x \in L(V)$ there exists $x' \in L(V)$ such that both $V(x) = V(x')$ and $|x'| \leq q(||V(x)||)$.

We define the following set A:

$$A = \{(\Pi, x) : \Pi \subseteq \Sigma^* \wedge x \in \Sigma^* \ \wedge \\ (\forall \pi \in \Sigma^* : \pi \leq p(|x|))[\pi \in V(x) \iff \pi \in \Pi]\}.$$

It is not hard to see that $A \in \Pi_i^p$ since $V \in \mathrm{P}^{\Sigma_{i-1}^p}$. Observe that the set A can be also written as $A = \{(\Pi, x) : \Pi \subseteq \Sigma^* \wedge x \in \Sigma^* \ \wedge\ V(x) = \Pi\}$. It follows that

$$\begin{aligned}\mathrm{INVS}_V &= \{\Pi \subseteq \Sigma^* : (\exists x \in \Sigma^*)[V(x) = \Pi]\}\\ &= \{\Pi \subseteq \Sigma^* : (\exists x \in \Sigma^* : |x| \leq q(||\Pi||))[V(x) = \Pi]\}\\ &= \{\Pi \subseteq \Sigma^* : (\exists x \in \Sigma^* : |x| \leq q(||\Pi||))[(\Pi, x) \in A]\}\end{aligned}$$

and thus $\mathrm{INVS}_V \in \Sigma^p_{i+1}$.

Even though inverting fair Σ^p_i-verifiers has, in general, an upper complexity bound of Σ^p_{i+1}, inversion of fair Σ^p_i-verifiers can be very easy in special cases.

Lemma 1. *For all $i \geq 1$ and all $B \in \Sigma^p_i$ there exists a fair Σ^p_i-verifier S such that* $\mathrm{INVS}_S \equiv^{\log}_m B$.

Proof. The proof is based on a proof given in [2]. Let B be a set from Σ^p_i and let R be a Σ^p_i-verifier such that $L(R) = B$. Consider the verifier S that is defined by $((x,\pi),x) \in S \leftrightarrow (x,\pi) \in R$ for all $x,\pi \in \Sigma^*$. Clearly, S is a Σ^p_i-verifier. It is straightforward to verify that S is also a fair Σ^p_i-verifier. Note that for all (x,π), the set $S((x,\pi))$ contains at most one proof, namely x itself.

We will now show that $x \in B \leftrightarrow \{x\} \in \mathrm{INVS}_S$ which yields the claim. First assume that $x \in B$ and thus there exists a certificate π such that $(x,\pi) \in R$ and thus $((x,\pi),x) \in S$. Since $S((x,\pi)) \subseteq \{x\}$ it follows that $S((x,\pi)) = \{x\}$. We conclude that $\{x\} \in \mathrm{INVS}_S$.

For the other direction assume that $x \notin B$ and hence for all $\pi \in \Sigma^*$ it holds that $(x,\pi) \notin R$ and thus $((x,\pi),x) \notin S$ for all certificates π. It follows that $S((x,\pi)) = \emptyset$ for all $\pi \in \Sigma^*$ which implies $\{x\} \notin \mathrm{INVS}_S$.

3.1 The Inverse Problem for Σ^p_i3CNFSAT

In the next two subsection we would like to examine the inverse complexity of natural verifiers for some selected complete problems in Σ^p_i. In particular we will look at the quantified versions of 3CNF-SAT and 3DNF-SAT and their natural verifiers.

Definition 7. An $i{+}1$-tuple $(F, X, Y_1, Y_2, \ldots, Y_{i-1})$ is called a type-i-formula if and only if F is a Boolean formula with variables from the set $X \cup Y_1 \cup ... \cup Y_{i-1}$ and $X, Y_1, ..., Y_{i-1}$ are pairwise disjoint sets. The set Σ^p_iSAT is defined as

$$\begin{aligned}\Sigma^p_i\mathrm{SAT} = \{\ (F, X, Y_1, \ldots, Y_{i-1}) : &(F, X, Y_1, \ldots, Y_{i-1}) \text{ is a type-}i\text{-formula} \quad \wedge\\ &(\exists \alpha \in \{0,1\}^{|X|})(\forall \beta_1 \in \{0,1\}^{|Y_1|}) \ldots\\ &(Q\, \beta_{i-1} \in \{0,1\}^{|Y_{i-1}|})[F(\alpha, \beta_1, ..., \beta_{i-1}) = 1]\}\end{aligned}$$

where $Q = \forall$ if i is even and $Q = \exists$ if i is odd. Here $F(\alpha, \beta_1, ..., \beta_{i-1})$ denotes the truth value of F when using α as a truth assignment for the variables of X and for all $1 \leq j \leq i-1$ using β_j as a truth assignment for the variables of Y_j.

It is well know that for all $i \geq 1$, the language Σ_i^pSAT is Σ_i^p-complete [10]. When restricting the formulas in Σ_i^pSAT to 3CNF or 3DNF formulas the set

$$\Sigma_i^p\text{3CNFSAT} = \{F : F \in \Sigma_i^p\text{SAT} \wedge F \text{ is a 3CNF-formula}\}$$

is Σ_i^p-complete for odd i's [10]. If i is even then the set

$$\Sigma_i^p\text{3DNFSAT}= \{F : F \in \Sigma_i^p\text{SAT} \wedge F \text{ is a 3DNF-formula}\}$$

is Σ_i^p-complete [10].

Let $i \geq 1$ be a natural number. The natural choice for a Σ_i^p-verifier for Σ_i^pSAT is certainly S_i, where

$$S_i(F,\alpha) \leftrightarrow (F, X, Y_1, \ldots, Y_{i-1}) \text{ is a type-}i\text{-formula} \wedge (\forall \beta_1 \in \{0,1\}^{|Y_1|})$$
$$(\exists \beta_2 \in \{0,1\}^{|Y_2|}) \ldots (Q\, \beta_{i-1} \in \{0,1\}^{|Y_{i-1}|})[F(\alpha, \beta_1, ..., \beta_{i-1}) = 1]$$

and $Q = \forall$ if i is even and $Q = \exists$ if i is odd. Analogously, the natural verifiers for Σ_i^p3CNFSAT and Σ_i^p3DNFSAT are C_i and D_i, where

$$(F,\alpha) \in C_i \leftrightarrow F \text{ is a type-}i\text{-formula in 3CNF} \wedge (F,\alpha) \in S_i,$$
$$(F,\alpha) \in D_i \leftrightarrow F \text{ is a type-}i\text{-formula in 3DNF} \wedge (F,\alpha) \in S_i,$$

and $Q = \forall$ if i is even and $Q = \exists$ if i is odd.

In this subsection we will concentrate on the inverse problem for the verifier C_i. In the next subsection we will proof results for the verifier D_i.

Lemma 2. *For all* $i \geq 1$ *it holds that* $\text{INVS}_{C_i} \subseteq \text{INVS}_{C_{i+1}}$.

Proof. The proof is obvious, since every type-i-formula $(F, X, Y_1, ..., Y_{i-1})$ in 3CNF has exactly the same satisfying assignments as the type-$i+1$-formula $(F, X, Y_1, ..., Y_{i-1}, Y_i)$ in 3CNF, where $Y_i = \emptyset$.

Next we will show a partial converse to Lemma 2, i.e., that $\text{INVS}_{C_1} = \text{INVS}_{C_2}$ (Theorem 3). Before formally stating and proving the theorem we will recall some helpful concepts and prove some lemmata.

Definition 8. [6]

1. Let $\Pi \subseteq \{0,1\}^n$ be a set of Boolean vectors, $\Pi = \{\pi_1, \pi_2, \ldots, \pi_k\}$. We call a Boolean vector $m \in \{0,1\}^n$ 3-compatible with Π if for any triple of indices (i_1, i_2, i_3), $1 \leq i_1 \leq i_2 \leq i_3 \leq n$, there exists a vector $\pi_j \in \Pi$ such that $m^{i_1} = \pi_j^{i_1}$ and $m^{i_2} = \pi_j^{i_2}$ and $m^{i_3} = \pi_j^{i_3}$.
2. A set $\Pi \subseteq \{0,1\}^n$ of Boolean vectors is called a 3CNF-set if and only if there is a 3CNF formula F such that the set of satisfying assignments of F is equal to Π.

Informally put, a vector $m \in \{0,1\}^n$ is 3-compatible with a set of Boolean vectors Π if and only for any sequence of three bit positions there exists a string in Π that agrees with m in these three positions.

The following Lemma from [6] gives a very tight connection between the notions of 3-compatibility and 3CNF-sets, namely, a set of Boolean vectors is a 3CNF-set if and only if it is closed under 3-compatibility.

Lemma 3. [6] *Let $\Pi \subseteq \{0,1\}^n$ be a set of assignments. Then Π is a 3CNF-set if and only if for all $m \in \{0,1\}^n$ that are 3-compatible with Π we have $m \in \Pi$.*

As an easy example consider the set $\Pi := \{0111, 1011, 1101, 1110\}$. The Boolean vector 1111 is 3-compatible with Π. But since $1111 \notin \Pi$ we conclude by Lemma 3 that there can not exist a 3CNF-formula F with exactly the satisfying assignments from Π.

Lemma 4. *Let $\Pi \subseteq \{0,1\}^n$ be a 3CNF-set. For all i, $1 \leq i \leq n$, and all $c \in \{0,1\}$ the set*

$$Cut^i_c(\Pi) := \{\ \alpha : \alpha \in \Pi\ \wedge\ \alpha^i = c\ \}$$

is a 3CNF-set.

Proof. Let $\Pi \subseteq \{0,1\}^n$ be a 3CNF-set, let $1 \leq i \leq n$, and $c \in \{0,1\}$. In order to show that $Cut^i_c(\Pi)$ is a 3CNF-set we use Lemma 3. We need to show that for all assignments α it holds that whenever α is 3-compatible with $Cut^i_c(\Pi)$ it also is an element of $Cut^i_c(\Pi)$. We will give a proof by contradiction.

So assume that $\alpha \in \{0,1\}^n$ is 3-compatible with $Cut^i_c(\Pi)$ yet $\alpha \notin Cut^i_c(\Pi)$. Hence $\alpha \notin \Pi$ or $\alpha^i \neq c$. We now argue that in both cases we have a contradiction. So assume that $\alpha \notin \Pi$. Since α is 3-compatible with $Cut^i_c(\Pi)$ it is also 3-compatible with any superset of $Cut^i_c(\Pi)$ and thus also 3-compatible with Π. However, by Lemma 3 we have that Π contains every assignment that is 3-compatible with Π, a contradiction. In case $\alpha^i \neq c$ we have an outright contradiction with the fact that α is 3-compatible with $Cut^i_c(\Pi)$. By definition for any three positions $1 \leq i_1 \leq i_2 \leq i_3 \leq n$ there exists a vector in $Cut^i_c(\Pi)$ that agrees with α in these three positions yet $\alpha^i \neq c$ and all vectors $\beta \in Cut^i_c(\Pi)$ satisfy $\beta^i = c$.

Lemma 5. *Let $\Pi \subseteq \{0,1\}^n$ be a 3CNF-set. For all $1 \leq i,j \leq n$ and all $c_1, c_2 \in \{0,1\}$ the set*

$$Cut^{i,j}_{c_1,c_2}(\Pi) := \{\ \alpha : \alpha \in \Pi\ \wedge\ (\ \alpha^i = c_1\ \vee\ \alpha^j = c_2\ )\ \}$$

is a 3CNF-set.

The proof is quite similar to the proof of Lemma 4 and thus omitted. We are now prepared to state and prove the main results of this section.

Theorem 3. $\mathrm{INVS}_{C_1} = \mathrm{INVS}_{C_2}$.

Proof. Due to Lemma 2 it suffices to show $\mathrm{INVS}_{C_2} \subseteq \mathrm{INVS}_{C_1}$.

Let $\Pi \in \mathrm{INVS}_{C_2}$. By definition of INVS_{C_2} we have $\Pi \neq \emptyset$ and there exists a type-2-formula (F, X, Y) in 3CNF over the variable set $X \cup Y$ of the form $F =$

$K_1 \wedge ... \wedge K_p$ where each K_i is a clause of the form $(z_1 \vee z_2 \vee z_3)$, $z_1, z_2, z_3 \in X \cup Y$, such that $C_2(F) = \Pi$. Recall that by definition of C_2 it holds that $(F, \alpha) \in C_2$ if and only if F is a type-2-formula in 3CNF and $(\forall \beta_1 \in \{0,1\}^{|Y|})F(\alpha, \beta) = 1$. In the remainder of this proof an assignment for a type-2-formula (F', X', Y') in 3CNF will be denoted by $\alpha\beta$, where α is the part of the assignment that assigns truth values to the variables from X' whereas β is the part of the assignment that assigns truth values to the variables from Y'.

We will now show that the set $C_2(F) = \Pi$ is itself a 3CNF-set and thus $\Pi \in \text{INVS}_{C_1}$. Observe that F does not contain a clause consisting solely of literals from the variable set Y since otherwise $C_2(F) = \Pi = \emptyset$, a contradiction. Hence, each clause of F contains at least one literal from the variable set X. We will construct a sequence of type-2-formulas in 3CNF $(F_0, X, Y), (F_1, X, Y), \ldots, (F_{n_1}, X, Y), (F_{n_1+1}, X, Y), \ldots, (F_{n_1+n_2}, X, Y)$ over the variable set $X \cup Y$ such that $C_2(F_{n_1+n_2}) = \Pi$. Indeed, we will prove by induction that for each $0 \leq i \leq n_1 + n_2$ the set $C_2(F_i)$ is a 3CNF-set.

Define F_0 to be the 3CNF formula that consists of all clauses from F that contain no literal from the variable set Y. Note that $C_2(F_0)$ is a 3CNF-set since F_0 is satisfied independent of assignments to the variables from Y. Let n_1 be the number of clauses in F that contain exactly one literal from the variable set Y. For each i, $0 \leq i \leq n_1 - 1$, let F_{i+1} be a type-2-formula in 3CNF such that $F_{i+1} = F_i \wedge K$ where K is a clause from F that contains exactly one literal from the variable set Y and K is not part of F_i. We will now argue that for all i, $1 \leq i \leq n_1$, $C_2(F_i)$ is a 3CNF-set. We will do this inductively. Recall that $C_2(F_0)$ is a 3CNF-set and assume that for some q, $1 \leq q \leq n_1$, $C_2(F_{q-1})$ is a 3CNF-set. Consider F_q. Let $F_q = F_{q-1} \wedge (\ell_i \vee \ell_j \vee \ell_k)$ where ℓ_i and ℓ_j are literals of the variables x_i and x_j, respectively, from X and ℓ_k is a literal of the variable y_k from Y. Observe that those assignments $\alpha\beta$ for F_q that (implicitly) assign the truth value 0 to ℓ_i, ℓ_j and ℓ_k can not satisfy F_q. It follows that no assignment α that assigns the truth value 0 to ℓ_i and ℓ_j can be in $C_2(F_q)$. On the other hand, any assignment from $C_2(F_{q-1})$ that assigns 1 to ℓ_i or ℓ_j or both is also in $C_2(F_q)$. Since trivially $C_2(F_{q-1}) \supseteq C_2(F_q)$ we have that $C_2(F_q) = Cut^{i,j}_{a,b}(C_2(F_{q-1}))$ where $a = 1$ if $\ell_i = x_i$ and $a = 0$ if $\ell_i = \overline{x_i}$ and similarly $b = 1$ if $\ell_j = x_j$ and $b = 0$ if $\ell_j = \overline{x_j}$. By Lemma 5 and the induction hypothesis we conclude that $C_2(F_q)$ is a 3CNF-set.

Let n_2 be the number of clauses in F that contain exactly two literals from the variable set Y. Similar to the above inductive argument related to clauses that contain exactly one literal from X one can easily show that $C_2(F_q) = Cut^{i}_{a}(C_2(F_{q-1}))$ for an appropriately chosen $a \in \{0,1\}$. By Lemma 4 and the induction hypothesis we have that $C_2(F_q)$ is a 3CNF-set.

To complete the proof observe that $F_q = F$ and thus $C_2(F) = \Pi$ is a 3CNF-set and hence $\Pi \in \text{INVS}_{C_1}$.

It has been shown in [6] that INVS_{C_1} is coNP-complete. Using the last theorem we have the following corollary.

Corollary 1. INVS_{C_2} *is* coNP*-complete.*

Next we will show that except the two coNP-complete problems INVS_{C_1} and INVS_{C_2} all other problems INVS_{C_i}, $i \geq 3$, are in P. We will do so by showing that for every syntactically correct set of assignments Π there exists a type-3-3CNF-formula with exactly the satisfying assignments from Π.

Theorem 4. *For all n and all $\Pi \subseteq \Sigma^n$ it holds that $\Pi \in \mathrm{INVS}_{C_3}$.*

Proof. Let $\Pi = \{\alpha_1, ..., \alpha_p\} \subseteq \Sigma^n$ for some $n \in \mathbb{N}$. In order to show $\Pi \in \mathrm{INVS}_{C_3}$ we will construct a type-3-3CNF-formula (F, X, Y_1, Y_2) over the variable sets X, Y_1, and Y_2 where $|X| = n$, $|Y_1| = 1$, and $|Y_2| = 2p + 1$ such that $C_3(F) = \Pi$.

We define an auxiliary set of assignments $\Pi' \subseteq \{0,1\}^{|X|+|Y_1|+|Y_2|}$ as follows:

$$\Pi' = \left\{ \begin{array}{c} \alpha_1 \ 0\ 000...00001, \\ \alpha_1 \ 1\ 000...00011, \\ \alpha_2 \ 0\ 000...00111, \\ \alpha_2 \ 1\ 000...01111, \\ \vdots \\ \alpha_p \ 0\ 001...11111, \\ \alpha_p \ 1\ 011...11111 \end{array} \right\} .$$

Claim: Π' is a 3CNF-set.

Proof of Claim: According to Lemma 3 it suffices to show that every assignment $\gamma \in \Sigma^{n+2p+2}$ that is 3-compatible with Π' is also an element of Π'.

So let $\gamma \in \Sigma^{n+2p+2}$ be an assignment that is 3-compatible with Π'. Hence it holds for any three positions k_1, k_2, and k_3, $1 \leq k_1 \leq k_2 \leq k_3 \leq n+2p+2$, that γ agrees with some $\gamma' \in \Pi'$ at these three positions. Since all assignments in Π' have a 0 at position $n+2$ and 1 at position $n+2p+2$ and since γ due to its 3-compatibility with Π' agrees with some assignment from Π' in particular at positions $n+2$, $n+2p+2$ and 1 it follows that $\gamma^{n+2} = 0$ and $\gamma^{n+2p+2} = 1$. Hence, there exists a position k, $n+2 \leq k \leq n+2p+1$, such that $\gamma^k = 0$ and $\gamma^{k+1} = 1$. Furthermore, for all positions k', $1 \leq k' \leq n+2p+2$, there exists an assignment $\gamma' \in \Pi'$ such that γ and γ' are equal at the positions k, $k+1$ and k'. However, there is only one assignment $\widehat{\gamma} \in \Pi'$ that has a 0 at position k and a 1 at position $k+1$. Hence $\widehat{\gamma}$ and γ have to agree at all positions k', $1 \leq k' \leq n+2p+2$. It follows that $\gamma = \widehat{\gamma}$ and thus $\gamma \in \Pi'$. This concludes the proof of the claim.

By the claim there exists a 3CNF-formula F' for which Π' is exactly the set of its satisfying assignments, $C_1(F') = \Pi'$. Let (F, X, Y_1, Y_2) denote a type-3-3CNF-formula where $F = F'$ and X, Y_1, and Y_2 are the sets of variables that correspond to the first n, the $n+1$st, and the last $2p+1$ truth values in each assignment in Π'. Now it is immediate that $C_3(F) = \Pi$.

Corollary 2. *For all n and all $\Pi \subseteq \Sigma^n$ it holds that a type-3-3CNF-formula (F, X, Y_1, Y_2) such that $\Pi = C_3(F)$ can be constructed in time polynomial in the size of Π.*

The proof is immediate from the proof of Theorem 4 and the fact that given a set of assignments a so called candidate formula for that set of assignments can be constructed in polynomial time [2].

Note that INVS_{C_3} already contains all possible syntactically correct proof sets Π for C_3, i.e., all proof sets where all certificates have the same length. To decide if Π belongs to INVS_{C_3} one therefore simply has to test if all certificates of Π have the same length, which can be tested in polynomial time in the size of Π. By Lemma 2 we furthermore have that for all $i \geq 3$ it holds that $\mathrm{INVS}_{C_i} = \mathrm{INVS}_{C_3}$.

Corollary 3. *For all $i \in \mathbb{N}$, $i \geq 3$, it holds*

1. $\mathrm{INVS}_{C_i} = \mathrm{INVS}_{C_3} = \{\Pi \subseteq \{0,1\}^* : (\exists n \in \mathbb{N})[\Pi \subseteq \{0,1\}^n]\}$.
2. $\mathrm{INVS}_{C_i} \in \mathrm{P}$.

Summarizing the results from this section, we can state that the inverse problems for the languages Σ_i^p3CNFSAT (based on their natural verifiers C_i) become easier with growing i.

3.2 The Inverse Problem for Σ_i^p3DNFSAT

In this subsection we will focus on the inverse problems related to Σ_i^p3DNFSAT as defined in Subsection 3.1.

We start by examining the problem Σ_1^p3DNFSAT. Note that Σ_1^p3DNFSAT belongs to P. Despite the fact that members of languages from P do not need any certificate, we feel that the verifier D_1 as defined in Section 3.1 is a natural verifier for Σ_1^p3DNFSAT. However, it is not immediately clear, that INVS_{D_1} is in P as well.

Theorem 5. $\mathrm{INVS}_{D_1} \in P$.

Proof. Let us first take a look at the structure of the proof set Π for a 3DNF-formula F. Let $F = \mathcal{M}_1 \vee ... \vee \mathcal{M}_m$ be a 3DNF-formula over the variable set $X = \{x_1, ..., x_n\}$ consisting of 3-monomials $\mathcal{M}_1, \dots, \mathcal{M}_m$. If $\mathcal{M} = (\ell_i \wedge \ell_j \wedge \ell_k)$, where $1 \leq i < j < k \leq n$ and either $\ell_t = x_t$ or $\ell_t = \overline{x_t}$ for all $t \in \{i, j, k\}$, is a monomial of the formula F then all assignments $\alpha \in \{0,1\}^n$ that assign the truth value 1 to ℓ_i, ℓ_j, and ℓ_k are satisfying assignments for the monomial $\mathcal{M}$. We denote the set of assignments for the formula F that satisfy the monomial $\mathcal{M}$ by $\Pi_{\mathcal{M}}$, i.e.,

$$\Pi_{\mathcal{M}} = \{\alpha \in \{0,1\}^n : \alpha \text{ as an assignment for } F \text{ satisfies } \mathcal{M}\}.$$

It is obvious that for the set of satisfying assignments Π of the formula F we have $\Pi = \Pi_{\mathcal{M}_1} \cup ... \cup \Pi_{\mathcal{M}_m}$.

In order to decide if a given set of assignments Π is contained in INVS_{D_1} we have to test if there exist 3-monomials $\mathcal{M}_1, ..., \mathcal{M}_m$ such that $\Pi = \Pi_{\mathcal{M}_1} \cup$

$... \cup \Pi_{\mathcal{M}_m}$. A deterministic polynomial-time algorithm for this decision problem works as follows: On input $\Pi \subseteq \{0,1\}^*$ test if there exists a natural number n such that $\Pi \subseteq \{0,1\}^n$. If so continue and otherwise reject the input Π. Next, test for each of the $8\binom{n}{3}$ possible 3-monomials $\mathcal{M}$ over the variable set $X = \{x_1, ..., x_n\}$ if $\Pi_{\mathcal{M}} \subseteq \Pi$. In case $\Pi_{\mathcal{M}} \subseteq \Pi$ mark all those assignments α in Π that are contained in $\Pi_{\mathcal{M}}$, otherwise continue with the next monomial. As a final step, check if there are unmarked assignments in Π and accept if this is not the case and reject otherwise.

Note that this algorithm runs in polynomial time in the size of Π. If all assignments are marked in the final stage of the algorithm it is immediate that the formula F, consisting of all 3-monomials $\mathcal{M}$ satisfying $\Pi_{\mathcal{M}} \subseteq \Pi$, has exactly the satisfying assignments from Π. If there is an unmarked assignment in Π then there exists no 3DNF-formula F such that $D_1(F) = \Pi$. This is since any unmarked assignment in Π has to be the satisfying assignment for a 3-monomial $\mathcal{M}$ that has additional assignments not contained in Π. This procedure can be accomplished in polynomial time in the size of Π.

A close look at the proof of Theorem 5 reveals that following corollary holds.

Corollary 4. *For all n and all $\Pi \subseteq \Sigma^n$ it holds that a 3DNF-formula F over n variables such that $\Pi = D_1(F)$ can be constructed in time polynomial in the size of Π.*

Regarding INVS_{D_i} for $i \geq 1$ we can in analogy to Lemma 2 state the following.

Lemma 6. *For all $i \geq 1$ it holds that* $\mathrm{INVS}_{D_i} \subseteq \mathrm{INVS}_{D_{i+1}}$.

Next we will introduce the main idea used in the proof of Theorem 6 at an easy example, otherwise the proof of Theorem 6 would become slightly intricate.

Lemma 7. *For all $\Pi \subseteq \{0,1\}^*$ with $|\Pi| = 1$ it holds that $\Pi \in \mathrm{INVS}_{D_2}$.*

Proof. Let $\Pi \subseteq \{0,1\}^*$ such that $|\Pi| = 1$ and let $\alpha \in \{0,1\}^n$ denote the single string contained in Π, i.e., $\Pi = \{\alpha\}$.

We will define a Σ_2^p-3DNF-formula (F, X, Y) with $D_2(F) = \{\alpha\}$, $X = \{x_1, x_2, \ldots, x_n\}$, and $Y = \{y_1, y_2, \ldots, y_{n-3}\}$. The formula F is defined as

$$\begin{aligned} F = (x_1^{\alpha^1} \wedge x_2^{\alpha^2} \wedge y_1) \ \vee \\ (\overline{y_1} \wedge x_3^{\alpha^3} \wedge y_2) \ \vee \\ (\overline{y_2} \wedge x_4^{\alpha^4} \wedge y_3) \ \vee \\ \ldots \vee \\ (\overline{y_{n-4}} \wedge x_{n-2}^{\alpha^{n-2}} \wedge y_{n-3}) \ \vee \\ (\overline{y_{n-3}} \wedge x_{n-1}^{\alpha^{n-1}} \wedge x_n^{\alpha^n}), \end{aligned}$$

where for any variable z, $z^0 = \overline{z}$ and $z^1 = z$. It remains to show that $D_2(F) = \{\alpha\}$.

First, observe that when assigning α to the variables from X the formula F is satisfied independent of the assignment of the variables from Y. Second, let $\alpha' \in \{0,1\}^n$, $\alpha \neq \alpha'$, be an assignment for the variables from X. Since $\alpha \neq \alpha'$ there exists $1 \leq i \leq n$ such that $\alpha^i \neq \alpha'^i$. However, it follows that the assignment $\alpha'\beta$, where $\beta^j = 0$ for all j smaller than $i-2$ and $\beta^j = 1$ for all other j, does not satisfy F.

This shows that the only assignment for the variables of X such that for all assignments of the variables from Y the formula F is satisfied is indeed α.

The main idea of the proof of Lemma 7 can be also used to prove the main result of this section. Similar to Theorem 4 we have that all syntactically correct set of proof are contained in INVS_{D_2}.

Theorem 6. *For all n and all $\Pi \subseteq \{0,1\}^n$ it holds that $\Pi \in \mathrm{INVS}_{D_2}$.*

Proof. Let $\Pi \subseteq \{0,1\}^n$, $\Pi = \{\alpha_1, \alpha_2, \ldots, \alpha_k\}$. Just as in the proof of Lemma 7 we will construct a formula F such that $D_2(F) = \Pi$. Informally, the formula F will consist of k subformulas in 3DNF $F_1, F_2, \ldots, F_k$ such that for all $1 \leq i \leq k$, $D_2(F_i) = \{\alpha_i\}$.

Let $X = \{x_1, x_2, \ldots, x_n\}$ and $Y = \{y_1, y_2, \ldots, y_{n-3}\}$ be disjoint sets. For each i, $1 \leq i \leq k$ we define a Σ_2^p-3DNF-formula (F_i, X, Y) as follows:

$$F_i = (x_1^{\alpha_i^1} \wedge x_2^{\alpha_i^2} \wedge y_1) \vee (\overline{y_1} \wedge x_3^{\alpha_i^3} \wedge y_2) \vee \ldots \vee \overline{y_{n-3}} \wedge x_{n-1}^{\alpha_i^{n-1}} \wedge x_n^{\alpha_i^n}).$$

The Σ_2^p-3DNF-formula (F, X, Y) is defined via $F = F_1 \vee F_2 \vee \ldots \vee F_k$. It follows from the proof of Lemma 7 that for each i, $1 \leq i \leq k$, there is exactly one assignment α for the variables of X, namely α_i, such that for all assignments β for the variables of Y, we have that $F_i(\alpha, \beta)$ is satisfied. It follows that $D_2(F) = \Pi$.

The proof of Theorem 6 contains an algorithm that given a set of assignments $\Pi \subseteq \Sigma^n$ constructs a type-2-3DNF-formula (F, X, Y) such that $D_2(F) = \Pi$. We hence have the following corollary.

Corollary 5. *For all n and all $\Pi \subseteq \Sigma^n$ it holds that a type-2-3DNF-formula (F, X, Y) such that $\Pi = D_2(F)$ can be constructed in time polynomial in the size of Π.*

In light of Lemma 6 we also have

Corollary 6. $\mathrm{INVS}_{D_i} \in \mathrm{P}$ *for all $i \geq 1$.*

Rounding off this section, we recall the verifier S_i ($i \in \mathbb{N}$) defined as

$$S_i(F,\alpha) \leftrightarrow (F,X,Y_1,\dots,Y_{i-1}) \text{ is a type-}i\text{-formula} \wedge (\forall \beta_1 \in \{0,1\}^{|Y_1|}) \\ (\exists \beta_2 \in \{0,1\}^{|Y_2|}) \dots (Q\,\beta_{i-1} \in \{0,1\}^{|Y_{i-1}|})[F(\alpha,\beta_1,...,\beta_{i-1}) = 1]$$

where $Q = \forall$ if i is even and $Q = \exists$ if i is odd. It can be seen as the natural verifier for the language Σ_i^pSAT.

Corollary 7. $\mathrm{INVS}_{S_i} \in \mathrm{P}$ *for all* $i \geq 1$.

The corollary again is obvious since every type-i-3DNF-formula is a type-i-formula.

Acknowledgments The authors would like to thank the anonymous referees for their very helpful comments.

References

1. Tobias Berg and Harald Hempel. Inverse problems have inverse complexity, 2008. Technical Report, FSU Jena.
2. Hubie Chen. Inverse NP problems. In *Mathematical foundations of computer science 2003*, volume 2747 of *Lecture Notes in Comput. Sci.*, pages 338–347. Springer, Berlin, 2003.
3. Michael R. Garey and David S. Johnson. *Computers and intractability.* W. H. Freeman and Co., San Francisco, Calif., 1979. A guide to the theory of NP-completeness, A Series of Books in the Mathematical Sciences.
4. Edith Hemaspaandra, Lane A. Hemaspaandra, and Harald Hempel. All superlinear inverse schemes are coNP-hard. *Theoret. Comput. Sci.*, 345(2-3):345–358, 2005.
5. Lane A. Hemaspaandra and Mitsunori Ogihara. *The complexity theory companion.* Texts in Theoretical Computer Science. An EATCS Series. Springer-Verlag, Berlin, 2002.
6. Dimitris Kavvadias and Martha Sideri. The inverse satisfiability problem. *SIAM J. Comput.*, 28(1):152–163 (electronic), 1999.
7. Michael Krüger and Harald Hempel. Inverse Hamiltonian cycle and inverse 3-D matching are coNP-complete. In *Algorithms and computation*, volume 4288 of *Lecture Notes in Comput. Sci.*, pages 243–252. Springer, Berlin, 2006.
8. Albert R. Meyer and Larry J. Stockmeyer. The equivalence problem for regular expressions with squaring requires exponential space. In *FOCS*, pages 125–129. IEEE, 1972.
9. Christos H. Papadimitriou. *Computational complexity.* Addison-Wesley Publishing Company, Reading, MA, 1994.
10. Larry J. Stockmeyer. The polynomial-time hierarchy. *Theoret. Comput. Sci.*, 3(1):1–22 (1977), 1976.