Inverse Circumscription

Hubie Chen
Department of Computer Science
Cornell University
Ithaca, NY 14853, USA
hubes@cs.cornell.edu

Abstract

Inverse (or *identification*) problems involve deciding whether or not an explicitly given set of data points have an implicit description, for instance, in the form of a constraint network. Such problems provide insight into the relationships among various representations of knowledge, which may have differing computational properties. This paper formalizes and studies the *inverse circumscription* problem, which (roughly speaking) is to decide, given a set of models, if there exists a formula whose circumscription describes the input set.

1 Introduction

The relationship between *implicit* and *explicit* descriptions of information is a central theme in knowledge representation and reasoning. For example, the core reasoning task of *propositional satisfiability* is to decide, given a propositional formula, whether or not the explicit set of models described implicitly by the formula is non-empty.

Recently, Kavvadias and Sideri [1998] studied a complementary task, which they called *inverse satisfiability*: given a set of models, is there a propositional formula with *exactly* the given set as its satisfying assignments? Without any restrictions on the class of formulas considered, the question is trivial, since for a given set of models M, it is always possible to create a formula in disjunctive normal form (DNF) capturing exactly M. Kavvadias and Sideri consider this question with respect to the class of formulas expressible by a prespecified *constraint language* – a set of relations which can be used to express constraints. Their intriguing main result is that the complexity of the inverse satisfiability problem reflects exactly that of the classical satisfiability problem: for a fixed constraint language, inverse satisfiability is intractable if and only if satisfiability is intractable! For instance, INVERSE 3-SAT is intractable, as 3-SAT is intractable; but, INVERSE 2-SAT is tractable, as 2-SAT is tractable.

The inverse satisfiability problem is akin to the *identification* problems studied by Dechter and Pearl [1992], which involve deciding whether or not explicitly given relational data in the form of tuples has an implicit description as a constraint network with particular features. Studying such *inverse* or *identification* problems is valuable for a variety of reasons, as articulated in [Dechter and Pearl, 1992; Kavvadias and Sideri, 1998]. First, certain restricted forms of propositional formulas, such as Horn formulas, facilitate efficient reasoning. Hence, the ability to decide when an explicitly given set of models can be represented in such a restricted form, is desirable. Second, model-based representations of information have been proposed as a viable alternative to formula-based representations [Kautz *et al.*, 1995; Khardon and Roth, 1996]; inverse problems address the relative expressibility of these two types of representations. Third, discovering structure in empirical data is a crucial component of scientific activity, and so understanding when structure discovery is computationally feasible sheds light on the nature of such activity.

In this paper, we further investigate the relationship between implicit and explicit descriptions of data by considering the complexity of *inverse circumscription*, an analog of the inverse satisfiability problem for circumscription. Circumscription is a well-studied non-monotonic reasoning formalism intended to embody common sense, and provides an alternative semantics for propositional formulas [McCarthy, 1980]. Intuitively, the models of the circumscription of a formula are those models of the original formula which make as few "assumptions" as necessary. The inverse circumscription problem addresses the expressiveness of circumscription: this problem involves deciding, roughly speaking, if a set of models can be described by a circumscribed formula (over some constraint language). Studying the complexity of this problem is natural not only for the aforementioned reasons, but also because circumscribed formulas are more *space-efficient* than uncircumscribed formulas: knowledge can be represented more succinctly with the former than with the latter, in a sense made precise in [Cadoli *et al.*, 1997].

There are at least two generalizations of the inverse satisfiability problem to the context of circumscription. One generalization is the problem of deciding, given a set of models, whether or not there exists a circumscribed formula (over a specified constraint language) with *exactly* the input set as its models. A second generalization is the problem of deciding, given a set of models, whether or not there exists a formula such that all models of the input set are models of the circumscribed formula – in other words, the problem is to decide the existence of an *approximating* formula whose circumscribed models covers the input set of models. We note that the tightness of the approximating formula is not what is at stake. In our formulation of this problem, it is possible to efficiently compute a candidate formula ψ such that if there is any formula ϕ covering the input set of models, then ψ also covers the input set *and* ψ has fewer models than ϕ (that is, the model set of ψ is a subset of the model set of ϕ).

As we will show, the complexity of the first generalization, for almost all constraint languages, can be characterized fairly easily using previous results. Consequently, the focus of this paper is on the complexity of the second generalization, which we call the inverse circumscription problem. Our main result is a *dichotomy theorem* in the complexity of inverse circumscription, which states that for *all* constraint languages, inverse circumscription is either in P, or is co-NP-complete.

Complexity dichotomy theorems are important because they demonstrate a full understanding of the complexity of a problem, with respect to a particular form of problem restriction (in our case, a restriction on the constraint language). The first dichotomy theorem was obtained by Schaefer, who showed that the propositional satisfiability problem over a fixed constraint language is either in P, or is NP-complete. The non-trivial tractable cases of satisfiability given by this result are 2-SAT, HORN SAT, and XOR-SAT (where each constraint is a linear equation in the field with two elements). Since Schaefer's result, many other dichotomy theorems have been established [Creignou *et al.*, 2001].

Kirousis and Kolaitis [2001a] very recently established a dichotomy theorem in the complexity of model checking for circumscription. The model checking problem is to decide, given an assignment and a formula, whether or not the assignment is a model of the circumscribed formula. Their theorem states that, for any constraint language, the model checking problem is either in P or is co-NP-complete. Interestingly, we show that (for any constraint language) the complexity of inverse circumscription is *exactly* the complexity of this model checking problem, giving a correspondence analogous to that between the complexity of inverse satisfiability and the complexity of satisfiability. In fact, to establish hardness of inverse circumscription, we will reduce from the model checking problem.

2 Preliminaries

In this section, we present the definitions and notation that will be used throughout the paper. We first introduce the notation of a constraint; constraints will be the building blocks of the propositional formulas we consider.

Definition 2.1 *A subset of* $\{0,1\}^k$ *(for some* $k \geq 1$*) is called a* logical relation*, and is said to have arity k.*

A constraint *over variable set V is a logical relation R paired with a list of variables* $x_1, \ldots, x_k \in V$ *where k is the arity of R, and is written* $R(x_1, \ldots, x_k)$.

A constraint with constants *over variable set V is a logical relation R and a list of variables or constants* $y_1, \ldots, y_k \in V \cup \{0,1\}$ *where k is the arity of R, and is written* $R(y_1, \ldots, y_k)$.

We next introduce the notion of a S-formula, which is a propositional formula built using the relations of a constraint language S as "templates."

Definition 2.2 *A* constraint language *S is a finite set of logical relations (which may contain relations of different arities).*[1] *The* maximum arity *of S is the maximum over the arities of the relations in S.*

*An S-*formula *over variable set V is a set of constraints (over V) where the logical relation of each constraint is in S.*

*An S-*formula with constants *over variable set V is a set of constraints with constants (over V) where the logical relation of each constraint is in S.*

We move on to describe the set of models associated with a formula. Intuitively, a model of a formula is an assignment to the variable set which obeys all constraints.

Definition 2.3 *Let* ϕ *be an S-formula (possibly with constants) over variable set V.*

An assignment to ϕ *is a function* $f : V \to \{0,1\}$. *When W is a subset of V,* $f_{|W}$ *denotes the restriction of f to W; and,* $\overline{f} : V \cup \{0,1\} \to \{0,1\}$ *denotes the extension of f where* $\overline{f}(0) = 0$, $\overline{f}(1) = 1$, *and* $\overline{f}(v) = f(v)$ *for all* $v \in V$.

The assignment f satisfies a constraint $R(x_1, \ldots, x_k)$ *of* ϕ *if the tuple* $(\overline{f}(x_1), \ldots, \overline{f}(x_k))$ *is in R.*

The assignment f satisfies *ϕ if it satisfies all constraints in ϕ; in this case, it is said to be a* satisfying assignment *or* model *of ϕ.*

Define Models(ϕ) to be the set containing all models of ϕ.

For a fixed S, the problem of deciding whether or not an input S-formula has a satisfying assignment is in NP, since in non-deterministic polynomial time, a satisfying assignment can be guessed and verified. The following example demonstrates that there is an S for which this satisfiability problem is equivalent to that for 3-SAT, and hence NP-complete.

Example 2.4 *Let* $S^3 = \{R_0, R_1, R_2, R_3\}$, *where* $R_0 = \{0,1\}^3 \setminus \{(0,0,0)\}$, $R_1 = \{0,1\}^3 \setminus \{(1,0,0)\}$, $R_2 = \{0,1\}^3 \setminus \{(1,1,0)\}$, *and* $R_3 = \{0,1\}^3 \setminus \{(1,1,1)\}$. *Every 3-SAT formula can be efficiently converted into a* S^3*-formula with exactly the same set of satisfying assignments, and vice-versa. For example, the 3-SAT formula*

$$(x \vee y \vee \neg z) \wedge (\neg w \vee y \vee \neg z) \wedge (z \vee \neg y \vee \neg w)$$

is equivalent to the S^3*-formula*

$$\{R_1(z,x,y), R_2(w,z,y), R_2(y,w,z)\}.$$

We generalize the definition of S^3 in Example 2.4 as follows.

Definition 2.5 *Let* S^k *(for* $k \geq 1$*) be the set of relations* $\{R_i^k : i \in \{0, \ldots, k\}\}$ *where* $R_i^k = \{0,1\}^k \setminus \{t_i^k\}$, *and* $t_i^k \in \{0,1\}^k$ *is the tuple with 1 in the first i coordinates, and 0 in the remaining* $k - i$ *coordinates.*

[1] Throughout this paper, we assume that all logical relations and constraint languages are non-empty.

Having defined the notion of a model of a formula, we now define what it means for a model to be minimal. The circumscription of a formula is considered to have, as its models, the minimal models of the original formula [McCarthy, 1980]. We consider minimality with respect to a subset P of the variable set: a model is minimal if there is no model "below" it in a comparison based on the variables in P.

Definition 2.6 *Let ϕ be an S-formula (possibly with constants) over variable set V, and let P be a subset of V. Let $\leq$ denote the standard total ordering on $\{0,1\}$ where $0 \leq 1$.*

Suppose that $\alpha : V \to \{0,1\}$ and $\beta : V \to \{0,1\}$ are both assignments over the same variable set V. We write $\alpha =_P \beta$ if for all $v \in P$, $\alpha(v) = \beta(v)$. We write $\alpha \leq_P \beta$ if for all $v \in P$, $\alpha(v) \leq \beta(v)$. We write $\alpha <_P \beta$ if $\alpha \leq_P \beta$ and it is not the case that $\alpha =_P \beta$. We write $\alpha \perp_P \beta$ if neither $\alpha \leq_P \beta$ nor $\beta \leq_P \alpha$ holds. When the set P is equal to V (the entire domain of α and β), we sometimes omit it when writing the above relations.

We say that β is a minimal satisfying assignment *(or* minimal model*) of ϕ with respect to P if β is a model of ϕ; and, for all models α of ϕ, $\alpha \leq_P \beta$ implies $\alpha =_P \beta$.*

Define MinModels(ϕ, P) to be the set containing all minimal models of ϕ with respect to P.

We now introduce some terminology that can be used to describe constraint languages; this terminology will prove to be quite wieldy in describing many of the complexity dichotomy theorems presented in the next section. As usual, we define a CNF-SAT formula over variable set V to be a conjunction of clauses, where each clause is the disjunction of literals from V. (A *literal* from variable set V is either a variable $v \in V$ itself, or the negation of a variable $\neg v$. Literals of the former type are called *positive*, whereas those of the latter type are called *negative*.) An assignment $f : V \to \{0,1\}$ satisfies a CNF-SAT formula ϕ if every clause of ϕ contains a literal evaluating to true under f.

Definition 2.7 *A bijunctive formula is a CNF-SAT formula where each clause contains exactly two literals.*

A (dual) Horn formula is a CNF-SAT formula where each clause contains at most one positive (negative) literal.

An affine formula is a conjunction of equations of the form $x_1 \oplus \cdots \oplus x_m = c$, where the x_i are variables and $c \in \{0,1\}$ is a constant. (The symbol $\oplus$ denotes the logical "exclusive or.")

Definition 2.8 *Let R be a logical relation of arity n, and S be a constraint language.*

The relation R is 0-valid *if it contains the all-zeroes tuple $(0,\dots,0)$ of arity n, and is* 1-valid *if it contains the all-ones tuple $(1,\dots,1)$ of arity n.*

The relation R is bijunctive (Horn, dual Horn, affine) if there exists a bijunctive (respectively Horn, dual Horn, affine) formula over $\{v_1,\dots,v_n\}$ with the same set of satisfying assignments as the $\{R\}$-formula $\{R(v_1,\dots,v_n)\}$.

The constraint language S is 0-valid (1-valid, bijunctive, Horn, dual Horn, affine) if every relation contained in S is 0-valid (respectively 1-valid, bijunctive, Horn, dual Horn, affine).

The constraint language S is Schaefer *if at least one of the following four conditions hold: S is bijunctive, S is Horn, S is dual Horn, S is affine. The constraint language S is* non-Schaefer *if it is not Schaefer.*

3 Related Work

This section reviews relevant work done previously; throughout, S is used to denote a constraint language. We first mention the seminal work of Schaefer, who proved a complexity dichotomy theorem on the satisfiability problem for S-formulas.

Definition 3.1 *The* $\mathrm{SAT}(S)$ *decision problem.*
Input: An S-formula ϕ.
Question: Is ϕ satisfiable?
The $\mathrm{SAT_C}(S)$ *decision problem is identical, except the input S-formula can have constants.*

In other words, the $\mathrm{SAT}(S)$ problem is to decide, for a given S-formula ϕ, if the set Models(ϕ) is non-empty. Remarkably, Schaefer proved that for *any* constraint language S, $\mathrm{SAT}(S)$ is either in P or is NP-complete; in addition, he gave a precise description of which constraint languages yield a tractable satisfiability problem, and which do not.

Theorem 3.2 *[Schaefer, 1978] Let S be a constraint language.*

If S is 0-valid, 1-valid, or Schaefer, then $\mathrm{SAT}(S)$ is in P; otherwise, $\mathrm{SAT}(S)$ is NP-complete.

If S is Schaefer, then $\mathrm{SAT_C}(S)$ is in P; otherwise, $\mathrm{SAT_C}(S)$ is NP-complete.

Kavvadias and Sideri [1998] studied the "inverse satisfiability problem" (denoted here by INVERSE $\mathrm{SAT_C}(S)$). In the "standard" satisfiability problem the goal is to decide, given a formula, whether or not there exists a satisfying assignment. In the "inverse" problem, the given input is a *set of assignments*, and the goal is to determine whether or not there exists a formula with exactly the given assignments as its set of satisfying assignments.

Definition 3.3 *The* INVERSE $\mathrm{SAT_C}(S)$ *decision problem.*
Input: Set A of assignments over the same variable set V.
Question: Is there an S-formula ϕ with constants such that $A =$ Models(ϕ)?

A dichotomy theorem was established by Kavvadias and Sideri, which shows that the inverse satisfiability problem is always in P, or is co-NP-complete. Intriguingly, the complexity of the inverse problem reflects *exactly* the complexity of the satisfiability problem: $\mathrm{SAT_C}(S)$ is intractable if and only if INVERSE $\mathrm{SAT_C}(S)$ is intractable!

Theorem 3.4 *[Kavvadias and Sideri, 1998] Let S be a constraint language.*

If S is Schaefer, then INVERSE $\mathrm{SAT_C}(S)$ *is in P; otherwise,* INVERSE $\mathrm{SAT_C}(S)$ *is co-NP-complete.*

We now formalize the model checking problem for circumscription, which was called "minimal satisfiability" in [Kirousis and Kolaitis, 2001a].

Definition 3.5 *The* MIN $\mathrm{SAT}(S)$ *decision problem.*
Input: An S-formula ϕ over variable set V, an assignment $\alpha : V \to \{0,1\}$ satisfying ϕ, and a subset P of V.

Question: Is the assignment α a minimal model of ϕ with respect to P?

The MIN SAT$_C(S)$ *decision problem is identical, except the input S-formula can have constants.*

The MIN SAT(S) problem is in co-NP, as deciding whether or not α is a minimal model of a formula amounts to verifying – for all assignments β – that if β is a model, then β is not strictly below α in the ordering $\leq_P$. Cadoli [1992] showed that this problem is co-NP-complete in general, and also identified some tractable cases. More recently, the following full dichotomy theorem was proved concerning the complexity of MIN SAT(S).

Theorem 3.6 *[Kirousis and Kolaitis, 2001a, Theorem 4.2]*[2]
Let S be a constraint language.
If S is Schaefer, then MIN SAT$_C(S)$ *is in P; otherwise,* MIN SAT$_C(S)$ *is co-NP-complete.*
If S is 0-valid or Schaefer, then MIN SAT(S) *is in P; otherwise,* MIN SAT(S) *is co-NP-complete.*

4 The Inverse Circumscription Problem

The inverse satisfiability problem (defined formally in the previous section) is to decide, given a set of assignments, whether or not there is a formula with precisely the given input set as its models. The *inverse circumscription problem*, denoted by INVERSE MIN SAT(S), is similar in that the input is also a set of assignments and the task is to decide whether or not there is a formula describing the input set. However, in the inverse circumscription problem, the question is whether there exists a formula ϕ such that all of the given assignments are *minimal* models of ϕ.

Definition 4.1 *The* INVERSE MIN SAT(S) *decision problem.*
Input: Set A of assignments over the same variable set V, and a subset P of V.
Question: Is there an S-formula ϕ such that $A \subseteq$ MinModels(ϕ, P)?
The INVERSE MIN SAT$_C(S)$ *decision problem is identical, except the question is to decide if there is an S-formula* with constants *satisfying the stated condition.*

As mentioned in the introduction, there is a natural variant of the INVERSE MIN SAT$_C(S)$ problem which has an identical description, except the $\subseteq$ symbol is replaced with an $=$ symbol in the "question." Let us denote this variant by INVERSE EXACT MIN SAT$_C(S)$. We have the following results concerning this question.

Theorem 4.2 *Let S be a constraint language.*
If S is bijunctive or Horn, then INVERSE EXACT MIN SAT$_C(S)$ *is in P.*
If S is non-Schaefer, then INVERSE EXACT MIN SAT$_C(S)$ *is co-NP-complete.*

For non-Schaefer constraint languages S, hardness of INVERSE EXACT MIN SAT$_C(S)$ can be shown by first establishing the hardness of INVERSE EXACT MIN SAT$_C(S^3)$ by reduction from INVERSE SAT$_C(S^3)$, and then reducing from INVERSE EXACT MIN SAT$_C(S^3)$ to INVERSE EXACT MIN SAT$_C(S)$, using a technique in [Kavvadias and Sideri, 1998]. For Schaefer constraint languages S, the existence of an output polynomial time algorithm for computing minimal models of a S-formula implies the tractability of INVERSE EXACT MIN SAT$_C(S)$; see [Kavvadias *et al.*, 2000] for such algorithms in the case of constraint languages S that are bijunctive or Horn.

For the remainder of this paper, we focus on the INVERSE MIN SAT(S) decision problem. Given an input set A of this problem, it is possible to efficiently compute a "candidate formula" having the property that if there is *any* formula containing A in its set of minimal models, then the candidate formula is such a formula. That is, the candidate formula serves to witness that "yes" is the answer to the decision question of Definition 4.1 – so long as some formula does.

Definition 4.3 *Suppose that S is a constraint language and that A is a set of assignments over the same variable set V. Define the* candidate S-formula for A *to be the set containing all constraints (over V and with relation in S) that are satisfied by every assignment in A. Similarly, define the* candidate S-formula for A with constants *to be the set containing all constraints with constants (over V and with relation in S) that are satisfied by every assignment in A.*

Lemma 4.4 *Suppose that S is a constraint language and that A is a set of assignments over a variable set V. Let P be a subset of V, and let ψ_C be the candidate S-formula (with constants) for A. There exists an S-formula (with constants) ϕ such that $A \subseteq$ MinModels(ϕ, P) if and only if $A \subseteq$ MinModels(ψ_C, P).*

For a fixed constraint language S, the candidate S-formula for a set of assignments A can be computed in polynomial time (measured with respect to the size of the representation of A). By the key property of the candidate formula (Lemma 4.4), it follows that INVERSE MIN SAT(S) is in co-NP: an assignment β which is not included in A, satisfies the S-candidate formula of A, and is strictly below an assignment in A (with respect to $\leq_P$), serves as a succinct and efficiently checkable proof that A is a "no" instance of INVERSE MIN SAT(S). (By similar reasoning, INVERSE SAT$_C(S)$ and INVERSE EXACT MIN SAT$_C(S)$ can be shown to be in co-NP, as discussed in [Kavvadias and Sideri, 1998].)

Whenever the model checking problem MIN SAT(S) is in P, the inverse problem INVERSE MIN SAT(S) will be in P. This is because deciding whether or not an input set A to the INVERSE MIN SAT(S) problem is a "yes" instance amounts to verifying that every assignment in A is a minimal model of the candidate formula for A; clearly, this can be done in polynomial time when MIN SAT(S) is in P. To prove a full dichotomy theorem on the complexity of

[2] We note that the notation of [Kirousis and Kolaitis, 2001a] is different from ours. In particular, their decision problem MIN SAT(S) involves checking, given an assignment and formula, if the assignment is minimal with respect to *all* variables. It is easily verified that our definition of MIN SAT$_C(S)$ is equivalent to their $(P;Q;Z)-$MIN SAT(S), and that our definition of MIN SAT(S) is equivalent to their $(P;\emptyset;Z)-$ MIN SAT(S). The theorem is stated here with respect to our notation.

INVERSE MIN SAT(S), it remains to describe the complexity of INVERSE MIN SAT(S) for the constraint languages S such that MIN SAT(S) is not in P.

5 Dichotomy Theorem

In this section, we describe completely the complexity profile of the "inverse circumscription problem." In particular, we show that for those constraint languages S such that MIN SAT(S) is co-NP-hard, INVERSE MIN SAT(S) is also co-NP-hard (and similarly for MIN $\text{SAT}_\text{C}(S)$ and INVERSE MIN $\text{SAT}_\text{C}(S)$).

Our first step is to prove hardness of the INVERSE MIN SAT(S) problem for 9-SAT formulas, where the circumscription is performed with respect to all of the variables. This initial hardness result is then used to establish the hardness of INVERSE MIN SAT(S) for other constraint languages S.

Theorem 5.1 *The problem* INVERSE MIN SAT(S^9) *is co-NP-hard, even with the restriction that the subset P must be equal to the entire variable set.*

The proof of this theorem is sketched in Appendix A; the hardness result claimed by the theorem is achieved by reduction from the MIN SAT(S) problem. The hardness of INVERSE MIN SAT(S^9) can then be leveraged to establish the hardness of INVERSE MIN $\text{SAT}_\text{C}(S)$, for the remaining constraint languages S.

Theorem 5.2 *Let S be a constraint language. If S is non-Schaefer, then* INVERSE MIN $\text{SAT}_\text{C}(S)$ *is co-NP-hard.*

Then, constants can be "removed" in such a way that allows the hardness of INVERSE MIN SAT(S) to be established, based on the hardness of INVERSE MIN $\text{SAT}_\text{C}(S)$.

Theorem 5.3 *Let S be a constraint language. If S is neither 0-valid nor Schaefer, then* INVERSE MIN SAT(S) *is co-NP-hard.*

Collecting together the theorems of this section as well as the discussion at the end of Section 4, we have the following dichotomy theorem.

Theorem 5.4 *Let S be a constraint language.*

If S is Schaefer, then INVERSE MIN $\text{SAT}_\text{C}(S)$ *is in P; otherwise,* INVERSE MIN $\text{SAT}_\text{C}(S)$ *is co-NP-complete.*

If S is 0-valid or Schaefer, then INVERSE MIN SAT(S) *is in P; otherwise,* INVERSE MIN SAT(S) *is co-NP-complete.*

6 Conclusions and Future Work

In this paper, we formalized and studied *inverse circumscription*. We established a full dichotomy theorem in the complexity of this problem (Theorem 5.4). A fascinating phenomenon is that the complexity of inverse circumscription reflects *exactly* the complexity of model checking for circumscription (Theorem 3.6). This correspondence in complexity parallels the intimate relationship between the complexity of inverse satisfiability (Theorem 3.4) and that of satisfiability (Theorem 3.2). The resemblance between the results on circumscription and those on propositional logic seems quite strong, as the hardness of inverse circumscription is established by reduction from model checking for circumscription – just as the hardness of inverse satisfiability is established by reduction from satisfiability [Kavvadias and Sideri, 1998].

It would be of great interest to investigate further the relationship between "inverse" problems (mapping an explicit description to an implicit description) and more classical "forward" problems (mapping an implicit description to an explicit description). A concrete goal for future work is to study inverse problems for other non-monotonic reasoning formalisms.

Acknowledgements. The author wishes to thank Bart Selman for useful discussions and suggestions, and Joe Halpern for his advice on the preparation of the final version of this paper.

A Proof Sketch of Theorem 5.1

Definition A.1 *Let A be a set of assignments over the variable set V. For $k \geq 1$, we say that an assignment $b : V \to \{0,1\}$ is k-compatible with A if for every size k subset W of V, there exists an assignment $a \in A$ such that $a_{|W} = b_{|W}$.*

Lemma A.2 *Let S be a constraint language with maximum arity k, and let A be a set of assignments over the variable set V. Suppose that ψ_C is the candidate S-formula for A (with $k \geq 1$). If b is r-compatible with A for all $r = 1, \ldots, k$, then b satisfies ψ_C. Moreover, the converse holds if $S = S^k$.*

Definition A.3 *Let S be a constraint language. For $r \geq 1$, an* r-pattern *of an S-formula ϕ over V is a pair (W, T) such that $|W| = r$, $W \subseteq V$, $T : W \to \{0,1\}$ is an assignment to the variables of W, and for all clauses $R(x_1, \ldots, x_k) \in \phi$, if T is defined on the variables of $R(x_1, \ldots, x_k)$ (that is, $\{x_1, \ldots, x_k\} \subseteq W$), then T satisfies $R(x_1, \ldots, x_k)$.*

In other words, an r-pattern for a formula ϕ is a subset W of the variable set of ϕ of size r along with an assignment to W which does not falsify any clause of ϕ.

Before giving the proof, we introduce the following notation. A vector $\vec{x}$ of length k is an ordered list of variables $(x_1, \ldots, x_k)$. If θ is a bit vector $(b_1, \ldots, b_k) \in \{0,1\}^k$ and $\vec{x}$ is a vector of length k, we let $\theta(\vec{x})$ denote the function mapping x_i to b_i for all $i = 1, \ldots, k$. If $\mu : V_1 \to \{0,1\}$ and $\nu : V_2 \to \{0,1\}$ are assignments with disjoint domains (that is, $V_1 \cap V_2 = \emptyset$), by $\mu \cup \nu$ we denote the function with domain $V_1 \cup V_2$ equal to μ on V_1, and equal to ν on V_2. When $\vec{z}$ is a vector of variables and f is an assignment defined on the variables of $\vec{z}$, we let $f_{|\vec{z}}$ denote the restriction of f to the variables of $\vec{z}$. For a positive integer k, we let $[k]$ denote the set $\{1, \ldots, k\}$.

Proof. By [Kirousis and Kolaitis, 2001a, Theorem 3.8], the version of MIN SAT(S^3) where the variable set P is promised to be the entire variable set is co-NP-complete. To prove hardness of INVERSE MIN SAT(S^9), we give a reduction from this version of MIN SAT(S^3). In particular, given a S^3-formula ϕ and an assignment a satisfying ϕ, we create a set of assignments A over a variable set X such that there is an S^9 formula ψ with $A \subseteq \text{MinModels}(\psi, X)$ if and only if a is a minimal model of ϕ. (By a minimal model of ϕ, we mean a minimal model with respect to the entire variable set of ϕ.)

We let $V = \{v_1, \ldots, v_n\}$ denote the variable set of ϕ, and we assume without loss of generality that $n > 9$. For all $i \in [n]$, let $\vec{x_i^1}$, $\vec{x_i^2}$, and $\vec{x_i^3}$ be vectors of length 3. For all $i \in [n]$ and $j \in [m]$, let $\vec{y_i^j}$ be a vector of length 2. Let X_i denote the set containing all $9 + 2m$ variables in the vectors $\vec{x_i^1}, \vec{x_i^2}, \vec{x_i^3}$ and $\vec{y_i^1}, \ldots, \vec{y_i^m}$. Let $X = \cup_{i \in [n]} X_i$.

Let $\alpha = (0,0,1)$, $\beta = (1,0,1)$, and $\gamma = (0,1,0)$. Let $\sigma = (0,1)$ and $\tau = (1,0)$. The essential properties of these vectors that we will use are that for any vector $\vec{z}$ of length 3, $\alpha(\vec{z}) < \beta(\vec{z})$, $\alpha(\vec{z}) \perp \gamma(\vec{z})$, and $\beta(\vec{z}) \perp \gamma(\vec{z})$; and, for any vector $\vec{z}$ of length 2, $\sigma(\vec{z}) \perp \tau(\vec{z})$.

For $i \in [n]$, define $\alpha_i : X_i \rightarrow \{0,1\}$ to be $\{\alpha(\vec{x_i^1}), \alpha(\vec{x_i^2}), \alpha(\vec{x_i^3})\} \cup \bigcup_{k \in [m]} \tau(\vec{y_i^k})$. Similarly, for $i \in [n]$, define $\beta_i : X_i \rightarrow \{0,1\}$ to be $\{\beta(\vec{x_i^1}), \beta(\vec{x_i^2}), \beta(\vec{x_i^3})\} \cup \bigcup_{k \in [m]} \tau(\vec{y_i^k})$. For $i \in [n]$ and $j \in [m]$, define $\gamma_i^j : X_i \rightarrow \{0,1\}$ to be $\{\gamma(\vec{x_i^1}), \gamma(\vec{x_i^2}), \gamma(\vec{x_i^3})\} \cup \{\sigma(\vec{y_i^j})\} \cup \bigcup_{k \in [m], k \neq j} \tau(\vec{y_i^k})$.

Let $(W_1, T_1), \ldots, (W_m, T_m)$ be all of the 9-patterns of ϕ. Our set of assignments A will be of size $m+1$: it will have one assignment $a^i : X \rightarrow \{0,1\}$ for each of the 9-patterns (W_i, T_i), as well as one assignment $\tilde{a} : X \rightarrow \{0,1\}$ encoding the assignment $a : V \rightarrow \{0,1\}$. The partial assignments $\{\gamma_i^j\}$, $\{\alpha_i\}$, and $\{\beta_i\}$ will be used as building blocks to define the assignments in A.

Define

$$a_i^j = \begin{cases} \gamma_i^j & \text{if } v_i \notin W_j \\ \alpha_i & \text{if } v_i \in W_j \text{ and } T_j(v_i) = 0 \\ \beta_i & \text{if } v_i \in W_j \text{ and } T_j(v_i) = 1 \end{cases}$$

Define $a^j = \cup_{i \in [n]} a_i^j$.

Also, for any assignment $f : V \rightarrow \{0,1\}$, define $\tilde{f} : X \rightarrow \{0,1\}$ to be the assignment $\bigcup_{i \in [n], f(v_i)=0} \alpha_i \cup \bigcup_{i \in [n], f(v_i)=1} \beta_i$.

Define A, the output of the reduction, to be the set $\{a^1, \ldots, a^m\} \cup \{\tilde{a}\}$.

Let us say that a pair of assignments $(d : X \rightarrow \{0,1\}, c : X \rightarrow \{0,1\})$ is a *violating pair* if the following four conditions are met: 1) $d \notin A$, 2) $c \in A$, 3) $d < c$, and 4) d is 9-compatible with A. Intuitively, a violating pair (d, c) is evidence that there is no S^9-formula ψ such that $A \subseteq \text{MinModels}(\psi, X)$.

More formally, observe that a violating pair exists if and only if there exists a $d \notin A$ and $c \in A$ such that $d < c$ and d satisfies the candidate S^9-formula ψ_C for A, by Lemma A.2. This occurs if and only if it is not the case that $A \subseteq \text{MinModels}(\psi_C, X)$, since (as is easily verified) the assignments in A are pairwise incomparable. By Lemma 4.4, it is not the case that $A \subseteq \text{MinModels}(\psi_C, X)$ if and only if for all S^9-formulas ψ, it is not the case that $A \subseteq \text{MinModels}(\psi, X)$. Hence, a violating pair exists if and only if there is no S^9-formula ψ such that $A \subseteq \text{MinModels}(\psi, X)$.

The remainder of the proof, omitted due to space constraints, establishes that a violating pair exists if and only if a is not a minimal model of ϕ. This implies the correctness of our reduction, as then we have that there exists no S^9-formula ψ such that $A \subseteq \text{MinModels}(\psi, X)$ if and only if a is not a minimal model of ϕ. $\square$

References

[Cadoli, 1992] Marco Cadoli. The Complexity of Model Checking for Circumscriptive Formulae. *Information Processing Letters*, 44(3):113-118, 1992.

[Cadoli *et al.*, 1997] Marco Cadoli, Francesco M. Donini, Marco Schaerf, and Riccardo Silvestri. On Compact Representations of Propositional Circumscription. *Theoretical Computer Science*, 182:183-202, 1997.

[Creignou *et al.*, 2001] Nadia Creignou, Sanjeev Khanna, and Madhu Sudan. *Complexity Classifications of Boolean Constraint Satisfaction Problems*. SIAM Monographs on Discrete Mathematics and Applications, SIAM, 2001.

[Dechter and Pearl, 1992] Rina Dechter and Judea Pearl. Structure identification in relational data. *Artificial Intelligence*, 58:237–270, 1992.

[Kautz *et al.*, 1995] Henry Kautz, Michael Kearns, and Bart Selman. Horn Approximations of Empirical Data. *Artificial Intelligence*, 74(1):129-145, 1995.

[Kavvadias and Sideri, 1998] Dimitris Kavvadias and Martha Sideri. The Inverse Satisfiability Problem. *SIAM Journal on Computing*, 28(1):152–163, 1998.

[Kavvadias *et al.*, 2000] Dimitris J. Kavvadias, Martha Sideri and Elias C. Stavropoulos. Generating all maximal models of a Boolean expression. *Information Processing Letters*, 74:157-162, 2000.

[Khardon and Roth, 1996] Roni Khardon and Dan Roth. Reasoning with Models. *Artificial Intelligence*, 87:187-243, 1996.

[Kirousis and Kolaitis, 2001a] Lefteris M. Kirousis and Phokion G. Kolaitis. The Complexity of Minimal Satisfiability Problems. In *Proceedings of the 18th Annual Symposium on Theoretical Aspects of Computer Science*, volume 2010 of *Lecture Notes in Computer Science*, pages 407–418. Springer, 2001. Full version at: Electronic Colloquium on Computational Complexity – (www.eccc.uni-trier.de/eccc), Report No. 82, 2000.

[Kirousis and Kolaitis, 2001b] Lefteris M. Kirousis and Phokion G. Kolaitis. A Dichotomy in the Complexity of Propositional Circumscription. In *Proceedings of the 16th Annual IEEE Symposium on Logic in Computer Science – LICS 2001*, pages 71–80, 2001.

[McCarthy, 1980] John McCarthy. Circumscription—A Form of Non-Monotonic Reasoning. *Artificial Intelligence*, 13:27-39, 1980.

[Schaefer, 1978] Thomas J. Schaefer. The complexity of satisfiability problems. In *Proc. 10th Annual ACM Symposium on Theory of Computing*, pages 216–226, 1978.