Working with ARMs: Complexity Results on Atomic Representations of Herbrand Models[1]

Georg Gottlob

Institut für Informationssysteme, Technische Universität Wien, A-1040 Vienna, Austria
E-mail: gottlob@dbai.tuwien.ac.at

and

Reinhard Pichler

Institut für Computersprachen, Technische Universität Wien, A-1040 Vienna, Austria
E-mail: reini@logic.at

Received October 18, 1999

An atomic representation of a Herbrand model (ARM) is a finite set of (not necessarily ground) atoms over a given Herbrand universe. Each ARM represents a possibly infinite Herbrand interpretation. This concept has emerged independently in different branches of computer science as a natural and useful generalization of the concept of finite Herbrand interpretation. It was shown that several recursively decidable problems on finite Herbrand models (or interpretations) remain decidable on ARMs.

The following problems are essential when working with ARMs: Deciding the equivalence of two ARMs, deciding subsumption between ARMs, and evaluating clauses over ARMs. These problems were shown to be decidable, but their computational complexity has remained obscure so far. The previously published decision algorithms require exponential space. In this paper, we prove that all mentioned problems are coNP-complete.

Key Words: complexity; Herbrand models; automated deduction; automated model building; knowledge representation; logic programming.

1. INTRODUCTION AND OVERVIEW OF RESULTS

1.1. *ARMs*

The value of using models within the field of automated reasoning is widely acknowledged; e.g., models can be used to represent some domain specific knowledge which may help to speed up the deduction process via semantic resolution. Moreover, the practical value of a theorem prover can be improved by constructing a countermodel rather than just giving the answer "No" if some input formula is found to be not a theorem. Consequently, automated model building has evolved as an important discipline in automated deduction, as recent publications demonstrate (cf. [3, 4, 7, 25], etc.).

For actual work with models, two prerequisites are essential, namely, an appropriate representation of models and the existence of (efficient) algorithms for certain decision problems like the evaluation of clauses in such models (cf. Section 1.3 below). In [7] (and, similarly, in [3, 4]), so-called ARMs (atomic representations of Herbrand models) were introduced as atom sets $\mathcal{A} = \{A_1, \ldots, A_n\}$ over some Herbrand universe H with the intended meaning that a ground atom over H evaluates to **T** in the model represented by $\mathcal{A}$, iff it is an instance of some atom $A_i \in \mathcal{A}$.

By their capability of representing a large (possibly infinite) set of ground atoms by a small finite atom set $\mathcal{A}$, ARMs can also be considered as a useful generic knowledge representation tool. Hence, problems related to the ones studied here also arise in database theory (cf. [27]). Independently, equivalent notions have been developed in other fields of computer science, such as in machine learning, where *implicit generalizations* are studied as a formal basis of learning from counterexamples (cf. [16]), or in the definition of a semantics-capturing negation in logic programming (cf. [9, 15]). In

[1] This work was supported by the Austrian Science Fund (FWF) Project Z29-INF. A short version of this paper was presented at LICS'99 (cf. [11]).

0890-5401/01 $35.00

fact, it is in this area where the idea of nonground models was first introduced (cf. works on the *s*-semantics of logic programs, e.g.; [6]). In functional programming, the problem of *completeness of definition* is studied (cf. [15]). In all these areas, decision problems similar to ours arise, even though each area recurs to a different terminology. Hence, our results are applicable in all of the mentioned areas.

The primary aim of this paper is a thorough complexity analysis of several problems which have to be solved when one actually wants to work with ARMs. The algorithms provided for these problems in [7] require exponential space and so does, in principle, the algorithm given in [16] for solving a related problem on implicit generalizations. The same complexity bound also applies to the method of [4], where these problems are solved via reduction to equational formulae over the term algebra. Our membership proofs will show that these algorithms can be significantly improved. On the other hand, our hardness results clearly point out the limit for such improvements.

1.2. *Some Basic Terminology*

For a background on terms, clauses, etc., see [2, 17]. We only recall the most relevant concepts in this paper. Let Σ be a finite set of constant symbols, function symbols, and predicate symbols. In this paper, we identify constant symbols with function symbols of arity 0. The *Herbrand universe* H over Σ is the algebra of ground terms that can be constructed from the symbols in Σ. An arbitrary (i.e., not necessarily ground) term over Σ is called an H-term.

A *substitution* is a mapping $\lambda : V \to T$ from a set of variables V to a set T of terms, s.t. $\lambda(v) \neq v$ only for finitely many $v \in V$. The set of variables $v \in V$ with $\lambda(v) \neq v$ is referred to as the *domain* of λ. The terms $\lambda(v)$ with $\lambda(v) \neq v$ are called the *range* of λ. λ is called a *ground substitution* (on V) if $\lambda(v)$ is a ground term for all $v \in V$. A substitution can be extended homomorphically to a mapping on terms or on clauses, respectively, in the obvious way; e.g., let $\lambda : V \to T$ be a substitution and let E be a term or a clause. Then by $E\lambda$ we denote the result of simultaneously replacing every variable x in E by the term $\lambda(x)$. E' is an *instance* of E, iff there exists a substitution σ with $E' = E\sigma$. Moreover, an instance E' of E is called *ground*, if E' contains no variables. An instance $E\sigma$ of E is called an *H-instance* of E, iff all terms in the range of σ are H-terms. E' is called an *H-ground instance* of E, if E' is a ground instance and an H-instance of E. Analogously to [7], we write $G_H(E)$ to denote the set of all H-ground instances of E.

Recall that in a Herbrand model $\mathcal{M}$, the interpretation of constant symbols and function symbols is fixed; i.e., they are interpreted "by themselves," so to speak. Hence, $\mathcal{M}$ is fully determined by the interpretation of the predicate symbols. In other words, there is a one-to-one correspondence between a Herbrand model $\mathcal{M}$ and the set of H-ground atoms that evaluate to $\mathbf{T}$ in $\mathcal{M}$. In the case of an ARM $\mathcal{A} = \{A_1, \ldots, A_n\}$, this set of H-ground atoms, which evaluate to $\mathbf{T}$ in the model represented by $\mathcal{A}$, corresponds to the H-ground instances of the atoms in $\mathcal{A}$. We write $\mathcal{M}_\mathcal{A}$ to denote this model, which is uniquely determined by the atom set $\mathcal{A}$.

In automated model building, the so-called H-subsumption plays an important role both in the model construction process itself and for the actual work with an already constructed model. The concept of H-subsumption was introduced in [7]. A related version of subsumption (namely, the so-called c-dissubsumption) can be found in the works of Caferra *et al.* (cf. [3, 4]). The definitions of (first-order) subsumption and H-subsumption are recalled below. Note that in these definitions clauses are considered as sets of atoms.

DEFINITION 1.1 (First-Order Subsumption). Let C and D be clauses and let $\mathcal{E}$ and $\mathcal{F}$ be clause sets. We say that C *subsumes* D (written as $C \leq_s D$) iff there exists a substitution ϑ s.t. $C\vartheta \subseteq D$.

Moreover, $\mathcal{E}$ *subsumes* D (written as $\mathcal{E} \leq_s D$) iff there exists a clause $E \in \mathcal{E}$ s.t. $E \leq_s D$). Likewise, $\mathcal{E} \leq_s \mathcal{F}$ iff every clause $F \in \mathcal{F}$ is subsumed by $\mathcal{E}$.

DEFINITION 1.2 (H-Subsumption). Let C and D be clauses and let $\mathcal{E}$ and $\mathcal{F}$ be clause sets. Moreover, let H be a Herbrand universe. We say that C *H-subsumes* D (written as $C \leq_{sH} D$) iff every H-ground instance of D is subsumed by C.

Moreover, $\mathcal{E}$ *H-subsumes* D (written as $\mathcal{E} \leq_{sH} D$) iff every H-ground instance of D is subsumed by some clause $E \in \mathcal{E}$. Likewise, $\mathcal{E} \leq_{sH} \mathcal{F}$ iff $\mathcal{E} \leq_{sH} F$ for every $F \in \mathcal{F}$. By $\mathcal{E} =_{sH} \mathcal{F}$ we denote that both relations $\mathcal{E} \leq_{sH} \mathcal{F}$ and $\mathcal{F} \leq_{sH} \mathcal{E}$ hold.

The following example from [23] illustrates the difference between these two concepts of subsumption.

EXAMPLE 1.1 (First-Order Subsumption vs H-Subsumption). Let $C = P(x, y) \vee Q(x) \vee Q(y)$ and $D = P(x, y) \vee Q(a) \vee Q(b)$. Then we have:

- $C \not\leq_s D$, since there is no substitution ϑ with domain $\{x, y\}$, s.t., on the one hand, $P(x, y)\vartheta \in D$ and, on the other hand, $Q(x)\vartheta \in D$ and $Q(y)\vartheta \in D$.
- For $H = \{a, b\}$, $C \leq_{sH} D$: Let $D\vartheta$ be an H-ground instance of D; i.e., $x\vartheta \in \{a, b\}$ and $y\vartheta \in \{a, b\}$. Then $C\vartheta \leq_s D\vartheta$.
- For $H = \{a, b, c\}$, $C \not\leq_{sH} D$: Consider the H-ground substitution $\vartheta = \{x \leftarrow c, y \leftarrow c\}$. Then the H-ground instance $D\vartheta$ of D is not subsumed by C.

Remark. Apart from the predicate symbols, there is a one-to-one correspondence between a signature Σ and the resulting Herbrand universe H. Hence, it is justified to refer to the terms that are built up from the function symbols and constant symbols of Σ as H-terms. Likewise, if the set of predicate symbols is clear from the context, then we may talk about a *clause over H* in order to refer to a clause that is built up from the symbols in Σ. Talking about "*H-subsumption*" rather than, e.g., "Σ-*subsumption*" may look a bit inaccurate. However, such is the standard terminology in the automated model building literature and we have decided to use this (slightly inaccurate) terminology, too.

1.3. *The Problems Studied Here*

When working with ARMs, efficient algorithms for the following problems are essential.

DEFINITION 1.3. The *MODEL-EQUIVALENCE* problem over a Herbrand universe H is defined as follows: Given two atom sets $\mathcal{A} = \{A_1, \ldots, A_n\}$ and $\mathcal{B} = \{B_1, \ldots, B_m\}$ over H, do $\mathcal{A}$ and $\mathcal{B}$ represent the same model; i.e., do they have the same set of H-ground instances? In this case we say that $\mathcal{A}$ and $\mathcal{B}$ are *equivalent*.

DEFINITION 1.4. The *CLAUSE-EVALUATION* problem over a Herbrand universe H is defined as follows: Given an atom set $\mathcal{A} = \{A_1, \ldots, A_n\}$ and a clause C over H, does C evaluate to **T** in the model $\mathcal{M}_{\mathcal{A}}$ represented by $\mathcal{A}$?

The TOTAL-COVER problem defined below will be useful for our complexity analysis. It is also interesting by itself since it corresponds to the completeness of definition problem in functional programming (cf. [15]). Slightly more general than TOTAL-COVER is the ATOM-H-SUBSUMPTION problem defined in [7], which also captures the emptiness problem of implicit generalizations given in [16]; i.e., given a term t and instances $t\vartheta_1, \ldots, t\vartheta_n$ of t, is every H-ground instance of t an instance of some $t\vartheta_i$? Note that this kind of question as to whether all H-ground instances of an expression are covered by a set of expressions arises quite naturally in the field of knowledge representation. Formal definitions of TOTAL-COVER and ATOM-H-SUBSUMPTION are given below:

DEFINITION 1.5. The TOTAL-COVER problem over a Herbrand universe H is defined as follows: Given an atom set $\mathcal{A} = \{P(\vec{t}_1), \ldots, P(\vec{t}_n)\}$ over H, is every H-ground atom $P(\vec{s})$ over H an instance of some $P(\vec{t}_i) \in \mathcal{A}$?

DEFINITION 1.6. The ATOM-H-SUBSUMPTION problem over a Herbrand universe H is defined as follows: Given atom sets $\mathcal{A} = \{A_1, \ldots, A_n\}$ and $\mathcal{B} = \{B_1, \ldots, B_m\}$ over H, is every H-ground instance $B_i\vartheta$ of every $B_i \in \mathcal{B}$ an instance of some atom $A_i \in \mathcal{A}$ (which is written as $\mathcal{A} \leq_{sH} \mathcal{B}$)?

The following example from [7] will help to illustrate these decision problems:

EXAMPLE 1.2. Denote by H the Herbrand universe built from constant a and function symbol f, i.e., $H = \{f^i(a) \mid i \geq 0\}$ (here $f^0(a)$ denotes a). Define the ARMs $\mathcal{A} = \{P(x, a)\}$ and $\mathcal{B} = \{P(a, a), P(f(x), a)\}$ over H.

$\mathcal{A}$ and $\mathcal{B}$ have the same set of H-ground instances, namely, $\{P(f^i(a), a) \mid i \geq 0\}$. We thus say that $\mathcal{A}$ and $\mathcal{B}$ are equivalent (cf. Definition 1.3). By Definition 1.2 on H-subsumption, we may also write $\mathcal{A} =_{sH} \mathcal{B}$.

In the model $\mathcal{M}_\mathcal{A}$ represented by $\mathcal{A}$ (or, equivalently, in the model $\mathcal{M}_\mathcal{B}$ represented by $\mathcal{B}$), the clause $\neg P(a,a) \vee \neg P(f(a),a) \vee P(f(f(f(a))),a)$ evaluates to **T**, while $\neg P(a,x) \vee P(f(x),\ y)$ evaluates to **F**, since, e.g., the ground instance $\neg P(a,a) \vee P(f(a),\ f(a))$ does. This can be seen as follows: $P(a,a)$ evaluates to **T**, since it is an H-ground instance of the atom $P(x,a) \in \mathcal{A}$. Hence, $\neg P(a,a)$ evaluates to **F**. Likewise, $sP(f(a),\ f(a))$ evaluates to **F**, since it is not an instance of the ARM.

Now let H' be the Herbrand universe built from constants $\{a,b\}$ and function symbol f, and define ARMs $\mathcal{A}' = \{P(x,a)\}$ and $\mathcal{B}' = \{P(a,a),\ P(f(x),a)\}$ in the same way as $\mathcal{A}$ and $\mathcal{B}$, but over the Herbrand universe H'. Then $\mathcal{A}'$ contains the H'-ground instances $\{P(f^i(a),a) \mid i \geq 0\} \cup \{P(f^i(b),a) \mid i \geq 0\}$. Note that $P(b,a)$ is not an H'-ground instance of $\mathcal{B}'$. Hence $\mathcal{A}$ and $\mathcal{B}$ are not equivalent. In particular, $\mathcal{A}'$ H'-subsumes $\mathcal{B}'$, while $\mathcal{B}'$ does not H'-subsume $\mathcal{A}'$. We thus write $\mathcal{A} \leq_{sH'} \mathcal{B}$ and $\mathcal{B} \not\leq_{sH'} \mathcal{A}$, respectively (cf. Definition 1.6).

Let $\mathcal{Q} \preceq_P \mathcal{R}$ denote that the decision problem $\mathcal{Q}$ can be reduced in polynomial time to the problem $\mathcal{R}$ (i.e., $\mathcal{Q}$ is "easier" than $\mathcal{R}$). The following theorem on the reducibility of the above-mentioned problems is not particularly deep. However, it is very convenient for the complexity analysis in the subsequent sections.

THEOREM 1.1. *Over any Herbrand universe H, the following chain of reducibility relations holds*: *TOTAL-COVER* $\preceq_P$ *MODEL-EQUIVALENCE* $\preceq_P$ *ATOM-H-SUBSUMPTION* $\preceq_P$ *CLAUSE-EVALUATION.*

Proof. For a reduction from TOTAL-COVER to MODEL-EQUIVALENCE, consider an arbitrary instance $\mathcal{A} = \{P(\vec{t}_1), \ldots, P(\vec{t}_n)\}$ of TOTAL-COVER. Let $\mathcal{B} = \{P(\vec{z})\}$ be another atom set, where $\vec{z}$ consists of pairwise distinct variables. Then every H-ground atom $P(\vec{s})$ is an instance of some $P(\vec{t}_i) \in \mathcal{A}$, iff $\mathcal{A}$ and $\mathcal{B}$ have the same set of ground instances. MODEL-EQUIVALENCE can be easily reduced to ATOM-H-SUBSUMPTION; namely, $\mathcal{A}$ and $\mathcal{B}$ have the same set of ground instances, iff both relations $\mathcal{A} \leq_{sH} \mathcal{B}$ and $\mathcal{B} \leq_{sH} \mathcal{A}$ hold. Finally, for a reduction from ATOM-H-SUBSUMPTION to CLAUSE-EVALUATION note that $\mathcal{A} \leq_{sH} \mathcal{B}$ holds, iff all unit clauses $B_j \in \mathcal{B}$ evaluate to **T** in $\mathcal{M}_\mathcal{A}$. ■

Recall from Definition 1.2 that H-subsumption is not necessarily restricted to atoms. We thus get the following decision problem:

DEFINITION 1.7. The CLAUSE-H-SUBSUMPTION problem over a Herbrand universe H is defined as follows: Given clause sets $\mathcal{C} = \{C_1, \ldots, C_n\}$ and $\mathcal{D} = \{D_1, \ldots, D_m\}$ over H, is every H-ground instance $D_j\vartheta$ of every clause $D_j \in \mathcal{D}$ subsumed by some clause $C_i \in \mathcal{C}$ (which is written as $\mathcal{C} \leq_{sH} \mathcal{D}$)?

In [23] it is shown that clausal H-subsumption is a very strong redundancy criterion, which may help to significantly speed up an automated theorem prover. Apart from the complexity of atomic H-subsumption, the general case of clausal H-subsumption has the number of permutations of literals as an additional source of complexity. Hence, the Π_2^p-hardness of this problem was shown in [23]. However, Π_2^p-membership in the case of an arbitrary Herbrand universe was left as an open question. In Section 6 we shall show that the Π_2^p-membership indeed holds.

Remark. All of the problems defined above depend on the *choice of a specific Herbrand universe*. For convenience, we have decided to consider the Herbrand universe H as arbitrary but fixed. So, in principle, we have to deal with a whole collection of decision problems, which are in a sense "parameterized" by H. Note, however, that we get exactly the same complexity results if the Herbrand universe H is considered as part of a problem instance. In particular, the membership proofs in Sections 5 and 6 for the case of a fixed Herbrand universe can be taken over literally to the case where the Herbrand universe is considered as part of the input.

The term **H**-subsumption (both for atoms and clauses) was introduced in [7] in order to distinguish this kind of subsumption over some Herbrand universe from the *usual notion of subsumption*, which does not depend on a specific Herbrand universe. Since no such confusion can arise with the other decision problems studied here, the "H" is not included in their names, e.g., in accordance with [7], we talk about MODEL-EQUIVALENCE and CLAUSE-EVALUATION rather than MODEL-H-EQUIVALENCE and CLAUSE-H-EVALUATION, respectively.

All of the problems studied here can be expressed as *validity problems of equational formulae* (i.e., first-order formulae with the syntactic equality "=" as the only predicate symbol) over the term algebra; e.g., let $\mathcal{A} = \{P(t_{11}, \ldots, t_{1k}), \ldots, P(t_{n1}, \ldots, t_{nk})\}$ be an instance of TOTAL-COVER and let $\{x_1, \ldots, x_l\}$ denote the set of variables in $\mathcal{A}$. Moreover, let $\{z_1, \ldots, z_k\}$ be a set of variables s.t. the x_i's and z_j's are pairwise distinct. Then $\mathcal{A}$ covers all H-ground atoms $P(s_1, \ldots, s_k)$, iff the formula $\forall(z_1, \ldots, z_k)\exists(x_1, \ldots, x_l) \bigvee_{i=1}^{n}(t_{i1} = z_1 \wedge \cdots \wedge t_{ik} = z_k)$ is valid. Hence, our results also apply to the corresponding classes of equational formulae.

1.4. *Overview of Results*

In summary, we prove the following complexity results in this paper:

coNP-completeness. The following problems are coNP-complete over any nontrivial Herbrand universe H: TOTAL-COVER, MODEL-EQUIVALENCE, ATOM-H-SUBSUMPTION and CLAUSE-EVALUATION.

Π_2^p-membership. The CLAUSE-H-SUBSUMPTION problem over any Herbrand universe is in Π_2^p. Hence, by the Π_2^p-hardness proven in [23], CLAUSE-H-SUBSUMPTION is Π_2^p-complete for any nontrivial Herbrand universe H.

1.5. *Structure of the Paper*

Section 1 was devoted to an introduction and overview of the main results. In Section 2, we shall recall some more basic definitions and results. The coNP-hardness of the TOTAL-COVER problem (and, hence, of all the other problems from Theorem 1.1) will be proved in Section 3. In Section 4 we provide a formalism for representing the complement of an ARM, which will then be used in Section 5 for proving the coNP-membership of the CLAUSE-EVALUATION problem (and, hence, of all the other problems from Theorem 1.1). In Section 6, we prove the Π_2^p-membership of clausal H-subsumption. Finally, in Section 7, the main results of this paper are summarized and some directions for future work are outlined.

2. PRELIMINARIES

2.1. *Expressions and Their Representation*

An *expression* is either a term or an atom. By $[E \mid p]$ we denote the *subexpression* in E at position p, where *positions* in E are defined as strings of integers as follows:

1. The empty string ε is a position in E, and $[E \mid \varepsilon] = E$.
2. Let F be a function symbol or a predicate symbol, and let $\alpha \geq 1$ denote the arity of F. Moreover, let p be a position in E with $[E \mid p] = F(t_1, \ldots, t_\alpha)$. Then, for every $q \in \{1, \ldots, \alpha\}$, $p \circ q$ (or simply "pq") is also a position in E and $[E \mid pq] = t_q$.

The *concatenation of positions* can easily be generalized to positions p and q of arbitrary length; i.e., if $p = p_1 \cdots p_k$ is a position in E with $[E \mid p] = s$ and $q = q_1 \cdots q_l$ is a position in s with $[s \mid q] = t$, then $p \circ q = pq = p_1 \cdots p_k q_1 \cdots q_l$ is also a position in E and $[E \mid pq] = t$ holds. A *subposition* of a position $p = p_1 \cdots p_k$ is a prefix $p_1 \cdots p_j$ with $0 \leq j \leq k$.

The *(term) depth of an expression* E is defined as the maximum length of the positions in E, i.e., $\tau(E) = \max(\{k \mid \exists p = p_1 \cdots p_k, \text{ s.t. } p \text{ is a position in } E\})$. For a set $\mathcal{E}$ of expressions, we define $\tau(\mathcal{E}) = \max(\{\tau(E) \mid E \in \mathcal{E}\})$.

A term t is called a *subterm* of an expression E, iff there exists a position p in E with $[E \mid p] = t$. By $\mathrm{Var}(E)$ we denote the set of variables which are subterms of the expression E. The *minimal depth of occurrence* of a subterm t of E is defined as the length of the shortest position p with $[E \mid p] = t$; i.e., $\tau_{\min}(E, t) = \min(\{k \mid \exists p = p_1 \cdots p_k, \text{ s.t. } p \text{ is a position in } E \text{ and } [E \mid p] = t\})$.

Remember that an expression E can be represented by a tree $T(E)$ in the following way: All internal nodes are labelled with function symbols of arity ≥ 1 (in the case of an atom E, the root node is labelled with a predicate symbol). The degree of an internal node (i.e., the number of child nodes of such a node) corresponds to the arity of the labelling symbol. The leaf nodes are labelled either by a constant or by a variable symbol. Since every node of the tree $T(E)$ corresponds to a unique position in the

expression E and vice versa, we shall identify nodes with their positions. The symbol labelling the node p corresponds to the leading symbol of $[E \mid p]$.

A *branch* is a path in $T(E)$ connecting the root with a leaf node. Every path connecting the root node with some node p is uniquely determined by p. In particular, a branch is uniquely determined by its leaf node. An *ordering* "$<$" on the nodes of $T(E)$ can be defined via lexicographical ordering on the positions or, equivalently, as a top-down, left-to-right ordering of the nodes. This ordering "$<$" can then be extended to branches by identifying every branch with its leaf node.

We define the *size of an expression* as the number of positions in E (or, equivalently, the number of nodes in the tree representation of E); i.e., $\text{size}(E) = |\{p \mid p \text{ is a position in } E\}|$. Note that there are many different ways of representing an expression, where the number of symbols required may even vary exponentially. However, the number of positions in E seems to be a natural measure for the size of an expression, since for the most common representations of expressions the number of symbols is linearly bounded in the number of positions. For example, in the case of a representation as a string of symbols, the number of additional symbols required (i.e., parentheses and commas) is bounded by $2 \times \text{size}(E)$.

The notions of depth, minimal depth of occurrence, size, etc., can be extended to clauses and to tuples of terms in the obvious way; e.g., for a negated atom $\neg A$, we define $\tau(\neg A) = \tau(A)$, $\tau_{\min}(\neg A, t) = \tau_{\min}(A, t)$, and $\text{size}(\neg A) = \text{size}(A)$. Moreover, for a clause C with literals $\{L_1, \ldots, L_n\}$, we set $\tau(C) = \max(\tau(L_i) \mid 1 \le i \le n\})$, $\tau_{\min}(C, t) = \min(\tau_{\min}(L_i, t) \mid 1 \le i \le n\})$, and $\text{size}(C) = \sum_{i=1}^{n} \text{size}(L_i)$.

Analogously, for a term tuple $\vec{t} = (t_1, \ldots, t_k)$, we define $\tau(\vec{t}) = \max(\{\tau(t_i) \mid 1 \le i \le k\})$ as the term depth and $\text{size}(\vec{t}) = \sum_{i=1}^{k} \text{size}(t_i)$ as its size.

We do not distinguish between a term and a term tuple of dimension 1. If f is a function symbol of arity k and $\vec{u} = (u_1, \ldots, u_k)$ is a k-tuple of terms, then we write $f(\vec{u})$ as a short-hand notation for the term $f(u_1, \ldots, u_k)$. For technical reasons, we also admit term tuples of dimension 0. Hence, we may write $f(\vec{u})$ to denote an arbitrary term with leading symbol f, even if f is a constant symbol (or, equivalently, a function symbol of arity 0). Moreover, we will not explicitly mention the dimension of a term tuple, if it is clear from the context; e.g., when writing $f(\vec{u})$, we implicitly assume that the dimension of $\vec{u}$ coincides with the arity of f.

In [7], the set BT_H of the so-called "*linear base terms over H*" was introduced; i.e., for the Herbrand universe H with signature Σ, BT_H is defined as $BT_H = \{f(\vec{x}) \mid f \text{ is a function symbol in } \Sigma \text{ with arity } \alpha \ge 0 \text{ and } \vec{x} \text{ is a vector of pairwise distinct variables of dimension } \alpha\}$. The set BT_H induces a partition of H in that every ground term $s \in H$ is an instance of exactly one term of BT_H. Namely, let f denote the leading symbol of $s \in H$. Then s is an instance of $f(\vec{x})$ and s is not an instance of any term in $BT_H - \{f(\vec{x})\}$.

2.2. *Equational Problems and Constrained Clauses*

Constrained clauses provide a significant increase in expressive power w.r.t. standard clauses: By using equations or disequations as constraints, it is possible to restrict the set of ground instances of a clause. This increased expressive power will be put to work in Section 4, when we search for a representation of the complement of an ARM. For the definition of constrained clauses, we follow the approach of Caferra *et al.* (cf. [4]), who in turn make use of equational problems in the sense of [5].

In [5], an *equational problem* is defined as a formula $\exists \vec{w}\, \forall \vec{y}\, \mathcal{P}(\vec{w}, \vec{x}, \vec{y})$, where $\mathcal{P}(\vec{w}, \vec{x}, \vec{y})$ is a quantifier-free formula with equality "$=$" as the only predicate symbol. A disequation $s \neq t$ is a short-hand notation for a negated equation $\neg(s = t)$. The trivially true problem is denoted by $\top$ and the trivially false one by $\bot$. By $\mathcal{P} \equiv \mathcal{Q}$ we denote that the equational problems $\mathcal{P}$ and $\mathcal{Q}$ are syntactically identical.

In the context of constrained clauses in the sense of [4], equational problems are only interpreted over the term algebra. A Herbrand interpretation over H is given through an H-ground substitution σ, whose domain coincides with the free variables of the equational problem. The trivial problem $\top$ evaluates to $\mathbf{T}$ in every interpretation. Likewise, $\bot$ always evaluates to $\mathbf{F}$. A single equation $s = t$ is validated by a ground substitution σ, if $s\sigma$ and $t\sigma$ are syntactically identical ground terms. The interpretation of the connectives $\neg$, $\wedge$, $\vee$, $\exists$, and $\forall$ is as usual. A ground substitution σ which validates an equational problem $\mathcal{P}$ is called a *solution* of $\mathcal{P}$.

In [4], constrained clauses (c-clauses, for short) are defined as pairs $[c : \mathcal{P}]$, where c is a clause and $\mathcal{P}$ is an equational problem. Intuitively, $[c : \mathcal{P}]$ denotes the set of ground clauses $c\sigma$, s.t. σ is a

solution of $\mathcal{P}$. In this paper, we are only interested in c-clauses as a powerful formalism for representing sets of ground clauses; e.g., in Section 4 we shall encounter sets of ground clauses which have a finite representation via c-clauses, but which are, in general, not finitely representable by standard clauses. Moreover, we only need c-clauses of the following restricted form:

DEFINITION 2.1 (Constrained Atoms). A constrained clause $[a : \bigwedge_{i=1}^{n} s_i \neq t_i]$ is called a *constrained atom* iff the clause part consists of a single atom and the constraint part is either the empty conjunction $\top$ or a quantifier-free conjunction of disequations s.t. all variables from the constraints occur in the atom.

For uniformity between standard clause logic and c-clause logic, we shall use the notation G_H from [7] in order to denote the set of H-ground instances also in the case of *constrained* atoms; i.e., $G_H([a : \mathcal{P}]) = \{a\sigma \mid \sigma \text{ is a solution of } \mathcal{P}\}$. Note that the set $G_H([a : \mathcal{P}])$ is nonempty iff the constraint part $\mathcal{P}$ has at least one solution. In the case of an infinite Herbrand universe H, the following lemma provides a criterion for testing the latter condition (for a proof, see [5, Appendix B, Lemma 1]).

LEMMA 2.1 (Disequations over an Infinite H). *Let $\mathcal{P}$ be a conjunction of disequations over an infinite Herbrand universe H. Then $\mathcal{P}$ has at least one solution, iff it contains no trivial disequation of the form $t \neq t$.*

3. coNP-HARDNESS

In this section we prove the coNP-hardness of the problems TOTAL-COVER, MODEL-EQUIVALENCE, ATOM-H-SUBSUMPTION, and CLAUSE-EVALUATION. To this end, we only need to show that the TOTAL-COVER problem is coNP-hard since, by Theorem 1.1, this is the "easiest" one of these problems.

THEOREM 3.1 (coNP-hardness of TOTAL-COVER). *Let H be an arbitrary Herbrand universe with at least two elements. Then the TOTAL-COVER problem over H is coNP-hard.*

Proof. In order to prove the coNP-hardness of the TOTAL-COVER problem, we reduce the well-known coNP-complete problem co-3SAT to it; i.e.,

- *Instance*. $(\{x_1, \ldots, x_k\}, E)$, s.t. $E = (l_{11} \vee l_{12} \vee l_{13}) \wedge \cdots \wedge (l_{n1} \vee l_{n2} \vee l_{n3})$ is a Boolean formula and the l_{ij}'s are literals over the propositional variables $\{x_1, \ldots, x_k\}$; i.e., every l_{ij} is either of the form x_γ or of the form $\neg x_\gamma$ for some $\gamma \in \{1, \ldots, k\}$.
- *Question*. Is the formula E unsatisfiable; i.e., does E evaluate to **F** in every truth assignment $\mathcal{I}$ on the propositional variables $\{x_1, \ldots, x_k\}$?

H contains at least two elements. Hence, there exists a constant a in the signature of H and an additional function symbol f with arity $\alpha \geq 0$. Recall from Section 2.1 the definition of the set BT_H of *linear base terms*. In particular, every ground term in H is an instance of exactly one term in BT_H. Moreover, BT_H contains the term a and a term of the form $f(y_1, \ldots, y_\alpha)$ for pairwise distinct variables y_i.

Now let $(\{x_1, \ldots, x_k\}, E)$ be an arbitrary instance of the co-3SAT problem. Without loss of generality we may assume that no clause $C_i = (l_{i1} \vee l_{i2} \vee l_{i3})$ contains a pair of complementary literals, since otherwise C_i is trivially true in every truth assignment and may therefore be deleted. Moreover, let P denote a predicate symbol of arity k and let $\{z_1, \ldots, z_k\}$ be a set of pairwise distinct variables. Then we can reduce this instance of the co-3SAT problem into the instance $\mathcal{A} = \mathcal{A}' \cup \mathcal{A}''$ of the TOTAL-COVER problem, where the atom sets $\mathcal{A}'$ and $\mathcal{A}''$ are defined as

$$\mathcal{A}' = \bigcup_{u \in BT'_H} \{P(u, z_2, \ldots, z_k), P(z_1, u, z_3, \ldots, z_k), \ldots, P(z_1, \ldots, z_{k-1}, u)\},$$

with $BT'_H = BT_H - \{a,\ f(y_1, \ldots, y_\alpha)\}$, and

$$\mathcal{A}'' = \{P(t_{11}, \ldots, t_{1k}), \ldots, P(t_{n1}, \ldots, t_{nk})\}$$

with

$$t_{ij} = \begin{cases} a & \text{if } x_j \in \{l_{i1}, l_{i2}, l_{i3}\} \\ f(y_1, \ldots, y_\alpha) & \text{if } \neg x_j \in \{l_{i1}, l_{i2}, l_{i3}\} \\ z_j & \text{otherwise.} \end{cases}$$

Of course, this transformation can be done in polynomial time. Therefore, it only remains to prove the following equivalence: The Boolean formula E is unsatisfiable $\Leftrightarrow$ Every H-ground atom $P(s_1, \ldots, s_k)$ is an instance of some atom in $\mathcal{A}' \cup \mathcal{A}''$.

Note that the arguments t_{ij} of the atoms in $\mathcal{A}''$ encode the kind of occurrence of the propositional variable x_j in the ith clause in the following way: a positive occurrence of x_j is represented by a, a negative occurrence is represented by $f(y_1, \ldots, y_\alpha)$, and a variable stands for no occurrence. The atoms in $\mathcal{A}''$ make sure that we do not have to care about terms with a leading symbol different from a and f. Then the above equivalence is shown as follows.

"$\Rightarrow$" Suppose that E is unsatisfiable and let $P(s_1, \ldots, s_k)$ be an arbitrary H-ground atom. We have to show that $P(s_1, \ldots, s_k)$ is an instance of some atom $P(t_{i1}, \ldots, t_{ik}) \in \mathcal{A}$. If $P(s_1, \ldots, s_k)$ has some argument with leading symbol different from a and f, then $P(s_1, \ldots, s_k)$ is a ground instance of some atom in $\mathcal{A}'$; i.e., suppose that s_j is a term of the form $g(v_1, \ldots, v_\beta)$ with $g \neq f$ and $g \neq a$, then $BT_H - \{a, f(y_1, \ldots, y_\alpha)\}$ contains a term $g(y_1, \ldots, y_\beta)$, where the y_i's are pairwise distinct variables. Moreover, $\mathcal{A}'$ contains the atom $P(z_1, \ldots, z_{j-1}, u, z_{j+1}, \ldots, z_k)$ with $u = g(y_1, \ldots, y_\beta)$ and, therefore, $P(s_1, \ldots, s_k)$ is clearly an instance of this atom. But then it only remains to consider the case where all arguments s_j of $P(s_1, \ldots, s_k)$ have either a or f as the leading symbol. To this end, we define the following truth assignment $\mathcal{I}$ on the propositional variables $\{x_1, \ldots, x_k\}$,

$$\mathcal{I}(x_j) = \begin{cases} \mathbf{F} & \text{if } s_j = a \\ \mathbf{T} & \text{otherwise.} \end{cases}$$

By assumption, there is some clause $l_{i1} \vee l_{i2} \vee l_{i3}$ which evaluates to $\mathbf{F}$ in $\mathcal{I}$; i.e., every literal $l_{i\gamma}$ with $\gamma \in \{1, 2, 3\}$ evaluates to $\mathbf{F}$. We claim that then $P(s_1, \ldots, s_k)$ is an instance of $P(t_{i1}, \ldots, t_{ik})$. By construction, every variable z_j in $P(t_{i1}, \ldots, t_{ik})$ occurs at most once. Hence, it suffices to show for every component j separately that s_j is an instance of t_{ij}. If t_{ij} is a variable z_j, then s_j is of course an instance of t_{ij}. It therefore only remains to consider the cases where t_{ij} is not a variable:

Case 1. If $t_{ij} = a$ then, by the problem transformation, $x_j \in \{l_{i1}, l_{i2}, l_{i3}\}$. Thus, x_j evaluates to $\mathbf{F}$ in $\mathcal{I}$ and, therefore, $s_j = a$ by the definition of $\mathcal{I}$.

Case 2. If $t_{ij} = f(y_1, \ldots, y_\alpha)$, then $\neq x_j \in \{l_{i1}, l_{i2}, l_{i3}\}$ holds and, therefore, x_j evaluates to $\mathbf{T}$ in $\mathcal{I}$. Hence, by the definition of $\mathcal{I}$, s_j has the leading symbol f and, thus, s_j is an instance of t_{ij}.

"$\Leftarrow$" Suppose that every H-ground atom $P(s_1, \ldots, s_k)$ is an instance of some atom in $\mathcal{A}$ and let $\mathcal{I}$ be an arbitrary truth assignment on $\{x_1, \ldots, x_k\}$. In order to show that there is some clause $l_{i1} \vee l_{i2} \vee l_{i3}$ in E which evaluates to $\mathbf{F}$ in $\mathcal{I}$, we consider the atom $P(s_1, \ldots, s_k)$, where each argument s_j is defined as follows:

$$s_j = \begin{cases} a & \text{if } \mathcal{I}(x_j) = \mathbf{F} \\ f(a, \ldots, a) & \text{otherwise.} \end{cases}$$

By assumption, every H-ground atom with predicate symbol P is an instance of some atom in $\mathcal{A}' \cup \mathcal{A}''$. Note that, by the above construction of s_j, there is no argument in $P(s_1, \ldots, s_k)$ with a leading symbol different from a and f. Hence, $P(s_1, \ldots, s_k)$ must be an instance of some atom $P(t_{i1}, \ldots, t_{ik}) \in \mathcal{A}''$. We claim that every literal $l_{i\gamma}$ with $\gamma \in \{1, 2, 3\}$ in the ith clause evaluates to $\mathbf{F}$ in $\mathcal{I}$.

Case 1. Suppose that $l_{i\gamma}$ is a positive literal, i.e., $l_{i\gamma} = x_j$ for some propositional variable x_j. Hence, by the problem transformation, $t_{ij} = a$ holds. But then also $s_j = a$ holds, since $P(s_1, \ldots, s_k)$ is an

instance of $P(t_{i1}, \ldots, t_{ik})$. Thus, by the construction of $P(s_1, \ldots, s_k)$, it follows that $l_{i\gamma} = x_j$ evaluates to **F** in $\mathcal{I}$.

Case 2. On the other hand, if $l_{i\gamma}$ is a negative literal of the form $l_{i\gamma} = \neg x_j$, then $t_{ij} = f(y_1, \ldots, y_\alpha)$ holds. Hence, $s_j = f(a, \ldots, a)$, since s_j must have the same leading symbol as t_{ij}. But then, by the construction of $P(s_1, \ldots, s_k)$, x_j evaluates to **T** in $\mathcal{I}$ and, therefore, $l_{i\gamma} = \neg x_j$ evaluates to **F** in $\mathcal{I}$. ■

Remark. As we have recently noticed, slightly different versions of this result have been independently published in [12–14]. The credit for this result thus goes to these authors.

Recall from Theorem 1.1 that TOTAL-COVER is the easiest problem studied here. Hence, the following result follows immediately from the above theorem.

COROLLARY 3.1 (coNP-hardness). *Let H be an arbitrary Herbrand universe with at least two elements. Then the problems TOTAL-COVER, MODEL-EQUIVALENCE, ATOM-H-SUBSUMPTION, and CLAUSE-EVALUATION over H are coNP-hard.*

4. THE COMPLEMENT OF AN ARM

The purpose of this section is to provide a formalism for representing the complement of an ARM via *constrained atoms* (cf. Definition 2.1). To this end, we define the complement of a k-tuple of H- terms $\vec{t} = (t_1, \ldots, t_k)$, i.e., the set of all ground k-tuples $\vec{s} \in H^k$ that are not instances of $\vec{t}$. The definition of the complement of a single atom and of an atom set over H will then be an easy task. But first let us introduce some additional notation.

It is sometimes convenient to group successive components of a term tuple together, e.g., for $i \in \{1, \ldots, n\}$, let $\vec{t}_i = (t_{i1}, \ldots, t_{ik_i})$. Then we may write $(\vec{t}_1, \ldots, \vec{t}_n)$ to denote the term tuple $(t_{11}, \ldots, t_{1k_1}, t_{21}, \ldots, t_{2k_2}, \ldots, t_{n1}, \ldots, t_{nk_n})$.

Analogously to *constrained atoms*, we can also define *constrained term tuples* as pairs $T = [\vec{t} : X]$, where T contains all ground instances $\vec{t}\sigma$ of $\vec{t}$, s.t. σ is a solution of X. For the sake of a uniform treatment of term tuples with or without constraints, respectively, we use the following notation: Let $\vec{u}$ be a constrained term of the form $[(u_1, \ldots, u_k) : X]$, and let f be a function symbol of arity k. Then we write $f(\vec{u})$ to denote the constrained term $[f(u_1, \ldots, u_k) : X]$. Likewise, we may write $P(\vec{u})$ for a constrained term tuple $\vec{u} = [(u_1, \ldots, u_k) : X]$ to denote the constrained atom $[P(u_1, \ldots, u_k) : X]$. Note that the constraints in a constrained term tuple $[\vec{t} : X]$ always refer to the variables occurring in $\vec{t}$. Hence it is a purely notational matter whether we attach the constraints to a term tuple as a whole or whether we consider the constraints as part of the variables occurring in $\vec{t}$. Hence, more generally, we shall also write $(\vec{u}_0, f_1(\vec{v}_1), \vec{u}_1, f_2(\vec{v}_2), \vec{u}_2, \ldots, f_l(\vec{v}_l), \vec{u}_l)$ no matter whether $(\vec{u}_0, \vec{v}_1, \vec{u}_1, \vec{v}_2, \vec{u}_2, \ldots, \vec{v}_l, \vec{u}_l)$ is simply a term tuple or whether it is a constrained term tuple of the form $[(u_{01}, \ldots, u_{0m_0}, v_{11}, \ldots, v_{1n_1}, u_{11}, \ldots, u_{1m_1}, \ldots, v_{l1}, \ldots, v_{ln_l}, u_{l1}, \ldots, u_{lm_l}) : X]$. In the latter case, $(\vec{u}_0, f_1(\vec{v}_1), \vec{u}_1, f_2(\vec{v}_2), \vec{u}_2, \ldots, f_l(\vec{v}_l), \vec{u}_l)$ is a short-hand notation for the constrained term tuple $[(u_{01}, \ldots, u_{0m_0}, f_1(v_{11}, \ldots, v_{1n_1}), u_{11}, \ldots, u_{1m_1}, \ldots, f_l(v_{l1}, \ldots, v_{ln_l}), u_{l1}, \ldots, u_{lm_l}) : X]$.

We are now ready to give a formal definition of the complement of a term tuple.

DEFINITION 4.1 (Complement of a Term Tuple). Let H be a Herbrand universe with signature Σ and let $\vec{t}$ be a k-tuple of H-terms with $k \geq 1$. Then we define the set $\mathrm{comp}_\Sigma(\vec{t})$ inductively as follows:

Case 1. If $\vec{t} = (t_1, \ldots, t_k)$ consists of variables only, then we define

$$\mathrm{comp}_\Sigma(\vec{t}) = \{[(z_1, \ldots, z_k) : z_i \neq z_j] \mid 1 \leq i < j \leq n,\ t_i \text{ and } t_j \text{ are identical,}\ \text{all components } t_\alpha \text{ with } \alpha < i \text{ are different from } t_i\ \text{and } (z_1, \ldots, z_k) \text{ is a vector of pairwise distinct variables.}\}$$

Case 2. Otherwise, let $\vec{t} = (\vec{r}_0, f_1(\vec{s}_1), \vec{r}_1, f_2(\vec{s}_2), \vec{r}_2, \ldots, f_l(\vec{s}_l), \vec{r}_l)$ with $l \geq 1$, where $\vec{r}_0, \ldots, \vec{r}_l$ are vectors of variables whose dimension may possibly be 0 and $f_1(\vec{s}_1), \ldots, f_l(\vec{s}_l)$ are the nonvariable components of $\vec{t}$. Then we define

$$\begin{aligned}\mathrm{comp}_\Sigma(\vec{t}) = \{(\vec{z}_0, w_1, \vec{z}_1, w_2, \vec{z}_2, \ldots, w_l, \vec{z}_l) \mid \\ \text{for every } \alpha \in \{0, \ldots, l\},\ \vec{z}_\alpha \text{ is a vector of pairwise distinct variables,} \\ \quad \text{s.t. } \vec{z}_\alpha \text{ and } \vec{r}_\alpha \text{ have the same dimension,} \\ \text{there exists exactly one } \beta \in \{1, \ldots, l\}, \text{ s.t. } w_\beta \in BT_H - \{f_\beta(\vec{x})\} \\ \quad \text{and for all } \gamma \in \{1, \ldots, l\} - \{\beta\},\ w_\gamma \text{ is a fresh variable} \\ \text{and } \vec{z}_0, w_1, \vec{z}_1, w_2, \ldots, \vec{z}_l \text{ are pairwise variable disjoint.}\} \\ \cup\ \{(\vec{u}_0,\ f_1(\vec{v}_1), \vec{u}_1,\ f_2(\vec{v}_2), \vec{u}_2, \ldots,\ f_l(\vec{v}_l), \vec{u}_l) \mid \\ (\vec{u}_0, \vec{v}_1, \vec{u}_1, \vec{v}_2, \vec{u}_2, \ldots, \vec{v}_l, \vec{u}_l) \in \mathrm{comp}_\Sigma((\vec{r}_0, \vec{s}_1, \vec{r}_1, \vec{s}_2, \vec{r}_2, \ldots, \vec{s}_l, \vec{r}_l))\}.\end{aligned}$$

For technical reasons, we also consider term tuples $\vec{t}$ of dimension 0. (Note that this allows us an easy inductive definition in Case 2 above without having to worry whether the term tuple $(\vec{r}_0, \vec{s}_1, \vec{r}_1, \ldots, \vec{s}_l, \vec{r}_l)$ has any components at all.) In this case, we simply set $\mathrm{comp}_\Sigma(\vec{t}) = \emptyset$.

DEFINITION 4.2 (Complement of an Atom). Let Σ be a signature consisting of predicate symbols, function symbols and constant symbols and let $P(\vec{t})$ be an atom over Σ. Then we define the set $\mathrm{comp}_\Sigma(P(\vec{t}))$ as

$$\begin{aligned}\mathrm{comp}_\Sigma(P(\vec{t})) = \{Q(\vec{z}) \mid Q \text{ is a predicate symbol in } \Sigma \text{ with } Q \neq P \\ \text{and } \vec{z} \text{ is a vector of pairwise distinct variables.}\} \\ \cup\ \{\mathrm{comp}_\Sigma(P(\vec{u})) \mid \vec{u} \in \mathrm{comp}_\Sigma(\vec{t})\},\end{aligned}$$

where $\mathrm{comp}_\Sigma(\vec{t})$ is defined according to Definition 4.1 above.

The definition of $\mathrm{comp}_\Sigma(P(\vec{t}))$ for an atom $P(\vec{t})$ is clear, given that Definition 4.1 really captures the complement of a term tuple. Intuitively, the elements of $\mathrm{comp}_\Sigma(\vec{t})$ are obtained as follows: Suppose that a tuple $\vec{s}$ of ground terms is *not* an instance of $\vec{t}$. Then this will be detected by a straightforward matching algorithm either by finding a nonvariable position p in $\vec{t}$, s.t. $[\vec{t} \mid p]$ and $[\vec{s} \mid p]$ have a different leading symbol, or by finding two variable positions p_1 and p_2 in $\vec{t}$, s.t. $[\vec{t} \mid p_1]$ and $[\vec{t} \mid p_2]$ are identical but $[\vec{s} \mid p_1]$ and $[\vec{s} \mid p_2]$ are not. Hence, if we consider the tree representation of $\vec{t}$, then we can reach the complement of $\vec{t}$ by "deviating" from this tree representation in the following way: If a node p is labelled by a function symbol f with arity $\alpha \geq 0$ and if we replace the whole subtree with root p by the tree corresponding to a term $g(\vec{v})$ for some function symbol $g \neq f$, then the resulting term tuple $\vec{s}$ is certainly not unifiable with $\vec{t}$. Hence, all ground instances of $\vec{s}$ are in the complement of $\vec{t}$. Likewise, suppose that a variable x occurs in two different places in $\vec{t}$. If we replace these two occurrences of x by two fresh variables z_1 and z_2 and, moreover, add the constraint $z_1 \neq z_2$, then again all ground instances of the resulting constrained term tuple are in the complement of $\vec{t}$. In case of a deviation at a nonvariable position in $\vec{t}$, the depth of this position corresponds to the recursion depth of the definition of $\mathrm{comp}_\Sigma(\vec{t})$, where finally a component of $f_\beta(\vec{u}_\beta)$ is replaced by a term $w_\beta \in BT_H - \{f_\beta(\vec{x})\}$. Likewise, the deviation at two variable positions in $\vec{t}$ corresponds to the base case in the definition of $\mathrm{comp}_\Sigma(\vec{t})$, where a constraint is added. The following example will illustrate these ideas.

EXAMPLE 4.1 (Tree Representation and Complement of an Atom). Let $\Sigma = \{P^3,\ Q^2,\ f^2,\ g^1, a^0\}$ denote a signature, where the arity of each symbol is given through the exponent, and let $A = P(f(x, a),\ f(y, x), x)$ be an atom over Σ.

The tree representation of A is depicted in Fig. 1. Moreover, the following atoms contain only ground instances from the complement of A:

- deviation at depth 0 is $Q(z_1, z_2)$;

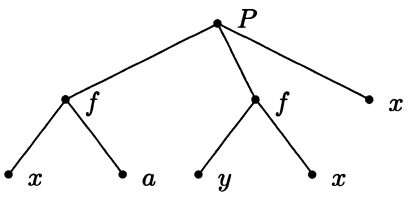

FIG. 1. Tree corresponding to $P(f(x, a),\ f(y, x), x)$.

- deviation at depth 1 is $P(a, y_1, z_1)$, $P(g(x), y_1, z_1)$, $P(y_1, a, z_1)$, $P(y_1, g(x), z_1)$;
- deviation at depth 2 is $P(f(z_1, g(x)), f(z_2, z_3), z_4)$, $P(f(z_1, f(x_1, x_2)), f(z_2, z_3), z_4)$;
- deviation at a variable position is $[P(f(z_1, a), f(z_2, z_3), z_4) : z_1 \neq z_3]$, $[P(f(z_1, a), f(z_2, z_3), z_4) : z_1 \neq z_4]$.

Now we show that $\mathrm{comp}_\Sigma(\vec{t})$ indeed captures the complement of a term tuple $\vec{t}$:

LEMMA 4.1 (Complement of a Term Tuple). *Let H be a Herbrand universe with signature Σ and let $\vec{t}$ be a k-tuple of H-terms. Then* $\mathrm{comp}_\Sigma(\vec{t})$ *from Definition* 4.1 *represents the complement of $\vec{t}$. That is, for any $\vec{s} \in H^k$, the following equivalence holds*: *$\vec{s}$ is not an instance of $\vec{t} \Leftrightarrow \vec{s}$ is an instance of a tuple in* $\mathrm{comp}_\Sigma(\vec{t})$.

Proof. We proceed by induction on the term depth $\tau(\vec{t})$ of $\vec{t}$.

Induction Begin. Let $\vec{t}$ be a term tuple with $\tau(\vec{t}) = 0$; i.e., the components of $\vec{t}$ are either constant symbols or variables. Then we distinguish the following cases.

Case 1. Let $\vec{t} = (t_1, \ldots, t_k)$ consist of variables only. For the "if"-direction, suppose that $\vec{s} = (s_1, \ldots, s_k)$ is an instance of some element $[(z_1, \ldots, z_k) : z_i \neq z_j]$ in $\mathrm{comp}_\Sigma(\vec{t})$. Then, in particular, the components s_i and s_j of $\vec{s}$ are distinct. On the other hand, t_i and t_j are two occurrences of the same variable in $\vec{t}$. Hence, for every ground instance $\vec{t}'$ of $\vec{t}$, the ith and jth components of $\vec{t}'$ must be identical. But then $\vec{s}$ cannot be an instance of $\vec{t}$.

For the "only if"-direction, suppose that $\vec{s}$ is not an instance of $\vec{t}$. Since $\vec{t}$ consists only of variables, there must be a pair (i, j) of components in $\vec{t}$ s.t. t_i and t_j are two occurrences of the same variable in $\vec{t}$ and the terms s_i and s_j are distinct. Without loss of generality, let (i, j) be the lexicographically smallest such pair. Note that $\vec{s}$ is of course an instance of the constrained tuple $[(z_1, \ldots, z_k) : z_i \neq z_j]$, where $(z_1, \ldots, z_k)$ is a vector of pairwise distinct variables. It only remains to prove that $[(z_1, \ldots, z_k) : z_i \neq z_j] \in \mathrm{comp}_\Sigma(\vec{t})$. By the definition of $\mathrm{comp}_\Sigma(\vec{t})$ in Case 1 of Definition 4.1, it suffices to show that $t_\alpha \neq t_i$ holds for all components t_α with $\alpha < i$. Suppose on the contrary that the variable t_i also occurs in some component t_α of $\vec{t}$ with $\alpha < i$. Then, by the condition $s_i \neq s_j$, we have either $s_\alpha \neq s_i$ or $s_\alpha \neq s_j$. Moreover, t_α and t_i as well as t_α and t_j are occurrences of the same variable in $\vec{t}$. Finally, (α, i) and (α, j) are lexicographically smaller than (i, j), which contradicts our assumption that (i, j) is the smallest such pair of components.

Case 2. Otherwise, $\vec{t} = (\vec{r}_0, a_1, \vec{r}_1, a_2, \vec{r}_2, \ldots a_l, \vec{r}_l)$ with $l \geq 1$, s.t. $\vec{r}_0, \ldots, \vec{r}_l$ are vectors of variables and $a_1, \ldots, a_l$ are constant symbols. Moreover, let $\vec{s} \in H^k$. For the "if" direction, suppose that $\vec{s}$ is an instance of some element in $\mathrm{comp}_\Sigma(\vec{t})$. Then we distinguish two possibilities:

1. $\vec{s}$ is an instance of a term tuple of the form $(\vec{z}_0, w_1, \vec{z}_1, \ldots w_l, \vec{z}_l)$ in $\mathrm{comp}_\Sigma(\vec{t})$ and there exist indices i and β, s.t. the ith component t_i of $\vec{t}$ is of the form $t_i = a_\beta$ and s_i is an instance of $w_\beta \in BT_H - \{a_\beta\}$. Hence, s_i and t_i are nonvariable terms with different leading symbols and, therefore, s_i cannot be an instance of t_i. But then $\vec{s}$ is not an instance of $\vec{t}$ either.

2. $\vec{s}$ is an instance of a term tuple of the form $(\vec{u}_0, a_1, \vec{u}_1, \ldots, a_l, \vec{u}_l)$ in $\mathrm{comp}_\Sigma(\vec{t})$, s.t. $(\vec{u}_0, \vec{u}_1, \ldots, \vec{u}_l)$ is in $\mathrm{comp}_\Sigma((\vec{r}_0, \vec{r}_1, \ldots, \vec{r}_l))$. Note that $(\vec{r}_0, \vec{r}_1, \ldots, \vec{r}_l)$ is a vector of variables. Hence, by Case 1 of Definition 4.1, there exist two occurrences of the same variable $r_{i\alpha_i}$ in $\vec{r}_i = (r_{i1}, \ldots, r_{in_i})$ and $r_{j\alpha_j}$ in $\vec{r}_j = (r_{j1}, \ldots, r_{jn_j})$, s.t. the constrained term tuple $(\vec{u}_0, \vec{u}_1, \ldots, \vec{u}_l)$ from $\mathrm{comp}_\Sigma((\vec{r}_0, \vec{r}_1, \ldots, \vec{r}_l))$ is of the form $(\vec{u}_0, \vec{u}_1, \ldots, \vec{u}_l) = [(z_{01}, \ldots, z_{0n_0}, \ldots, z_{l1}, \ldots, z_{ln_l}) : z_{i\alpha_i} \neq z_{j\alpha_j}]$. Thus, $\vec{s}$ is an instance of the constrained term tuple $[(z_{01}, \ldots, z_{0n_0}, a_1, z_{11}, \ldots, z_{1n_1}, \ldots, a_l, z_{l1}, \ldots, z_{ln_l}) : z_{i\alpha_i} \neq z_{j\alpha_j}]$ from $\mathrm{comp}_\Sigma(\vec{t})$. On the other hand, the components $r_{i\alpha_i}$ and $r_{j\alpha_j}$ in $\vec{t} = (r_{01}, \ldots, r_{0n_0}, a_1, r_{11}, \ldots, r_{1n_1}, \ldots, a_l, r_{l1}, \ldots, r_{ln_l})$ are two occurrences of the same variable. But then, analogously to the considerations in Case 1 above, $\vec{s}$ cannot be an instance of $\vec{t}$.

For the "only if" direction, suppose that $\vec{s}$ is not an instance of $\vec{t} = (r_{01}, \ldots, r_{0n_0}, a_1, r_{11}, \ldots, r_{1n_1}, \ldots, a_l, r_{l1}, \ldots, r_{ln_l})$. Then there are again two possibilities:

1. There exists a component a_i in $\vec{t}$ s.t. the corresponding component $f_i(\vec{u}_i)$ of $\vec{s}$ has a different leading symbol; i.e., $a_i \neq f_i$. Then $\vec{s}$ is an instance of the term tuple $T = (\vec{z}_0, w_1, \vec{z}_1, \ldots w_l, \vec{z}_l)$, where every $\vec{z}_\alpha$ is a vector of pairwise distinct variables, $w_i \in BT_H - \{a_i\}$, and for all $\gamma \in \{1, \ldots, l\} - \{i\}$, w_γ is a fresh variable. Moreover, by Case 2 of Definition 4.1, T is contained in $\mathrm{comp}_\Sigma(\vec{t})$.

2. $\vec{s}$ coincides with $\vec{t}$ on all nonvariable components a_i of $\vec{t}$ but there exist two occurrences $r_{i\alpha_i}$ and $r_{j\alpha_j}$ of the same variable in $\vec{t}$ s.t. the terms in the corresponding components of $\vec{s}$ are distinct. Without loss of generality, let $(i, \alpha_i, j, \alpha_j)$ be the lexicographically minimal quadruple with this property. Then $\vec{s}$ is of course an instance of the constrained term tuple $T = [(z_{01}, \ldots, z_{0n_0}, a_1, z_{11}, \ldots, z_{1n_1}, \ldots, a_l, z_{l1}, \ldots, z_{ln_l}) : z_{i\alpha_i} \neq z_{j\alpha_j}]$, where the components $z_{\beta\gamma}$ are pairwise distinct variables. Moreover, analogously to Case 1 above, it can be shown that $[(z_{01}, \ldots, z_{0n_0}, \ldots, z_{l1}, \ldots, z_{ln_l}) : z_{i\alpha_i} \neq z_{j\alpha_j}]$ is an element of $\mathrm{comp}_\Sigma((\vec{r}_0, \vec{r}_1, \ldots, \vec{r}_l))$. Hence, by Case 2 of Definition 4.1, T is contained in $\mathrm{comp}_\Sigma(\vec{t})$.

Induction Step. Let $d \geq 0$ and suppose that $\mathrm{comp}_\Sigma(\vec{u})$ indeed represents the complement of $\vec{u}$ for all term tuples $\vec{u}$ with $\tau(\vec{u}) \leq d$. Moreover, let $\vec{t}$ be a term tuple with $\tau(\vec{t}) = d + 1$. Then $\vec{t}$ is of the form $\vec{t} = (\vec{r}_0, f_1(\vec{s}_1), \vec{r}_1, f_2(\vec{s}_2), \vec{r}_2, \ldots, f_l(\vec{s}_l), \vec{r}_l)$ with $l \geq 1$, where $\vec{r}_0, \ldots, \vec{r}_l$ are vectors of variables and $f_1(\vec{s}_1), \ldots, f_l(\vec{s}_l)$ are the nonvariable components of $\vec{t}$. Now let $\vec{s}$ be an arbitrary ground term tuple from H^k with $\vec{s} = (\vec{r}_0{}', g_1(\vec{s}_1{}'), \vec{r}_1{}', g_2(\vec{s}_2{}'), \vec{r}_2{}', \ldots, g_l(\vec{s}_l{}'), \vec{r}_l{}')$, s.t. every $\vec{r}_i{}'$ has the same dimension as $\vec{r}_i$. Then the ground term $g_i(\vec{s}_i{}')$ is unifiable with $f_i(\vec{s}_i)$, iff g_i and f_i are identical and $\vec{s}_i{}'$ is an instance of $\vec{s}_i$. Likewise, $\vec{s}$ is unifiable with $\vec{t}$, iff for all $i \in \{1, \ldots, l\}$, g_i and f_i are identical and $S = (\vec{r}_0{}', \vec{s}_1{}', \vec{r}_1{}', \vec{s}_2{}', \vec{r}_2{}', \ldots, \vec{s}_l{}', \vec{r}_l{}')$ is an instance of $T = (\vec{r}_0, \vec{s}_1, \vec{r}_1, \vec{s}_2, \vec{r}_2, \ldots, \vec{s}_l, \vec{r}_l)$. In other words, $\vec{s}$ is in the complement of $\vec{t}$, iff either (1) there exists a nonvariable component $f_i(\vec{s}_i)$ in $\vec{t}$, s.t. the corresponding component $g_i(\vec{s}_i{}')$ in $\vec{s}$ has a different leading symbol, or (2) the leading symbol of every nonvariable component in $\vec{t}$ coincides with the corresponding component in $\vec{s}$ but S is not an instance of T. Analogously to Case 2 of Induction Begin, it can be shown that the ground instances of the tuples of the form $(\vec{z}_0, w_1, \vec{z}_1, w_2, \vec{z}_2, \ldots w_l, \vec{z}_l)$ from $\mathrm{comp}_\Sigma(\vec{t})$ are exactly those ground term tuples which fulfill condition (1) above.

On the other hand, the term depth of the tuple $T = (\vec{r}_0, \vec{s}_1, \vec{r}_1, \ldots, \vec{s}_l, \vec{r}_l)$ is strictly smaller than $\tau(\vec{t})$. Hence, by the induction hypothesis, we may conclude that the set of ground instances of the tuples in $\mathrm{comp}_\Sigma(T)$ coincides with the complement of T. But then also the ground instances of the tuples in $\mathcal{T} = \{(\vec{u}_0, f_1(\vec{v}_1), \vec{u}_1, \ldots, f_l(\vec{v}_l), \vec{u}_l) \mid (\vec{u}_0, \vec{v}_1, \vec{u}_1, \ldots, \vec{v}_l, \vec{u}_l) \in \mathrm{comp}_\Sigma(T)\}$ are exactly those ground term tuples which fulfill condition (2) above.

The only point that we have to be a bit careful about is that a term tuple $(\vec{u}_0, \vec{v}_1, \vec{u}_1, \ldots, \vec{v}_l, \vec{u}_l)$ in $\mathrm{comp}_\Sigma(T)$ may contain constraints, which are then also contained in the corresponding term tuple $(\vec{u}_0, f_1(\vec{v}_1), \vec{u}_1, \ldots, f_l(\vec{v}_l), \vec{u}_l)$ in $\mathcal{T}$. However, removing a function symbol f_i from a component in $\vec{t}$ and lifting the arguments $\vec{s}_i$ to components of $\vec{t}$ clearly preserve all the occurrences of the variables from $\vec{t}$. Likewise, when we produce a tuple $(\vec{u}_0, f_1(\vec{v}_1), \vec{u}_1, \ldots, f_l(\vec{v}_l), \vec{u}_l)$ by adding a function symbol f_i in front of the components $\vec{v}_i$ of the corresponding tuple in $\mathrm{comp}_\Sigma(T)$, then no variable occurrences are deleted or added. Hence, analogously to Case 2 of Induction Begin, it is a purely notational matter whether we consider the constraints of a tuple $(\vec{u}_0, f_1(\vec{v}_1), \vec{u}_1, \ldots, f_l(\vec{v}_l), \vec{u}_l)$ in $\mathcal{T}$ as part of the tuple itself or as part of the tuple $(\vec{u}_0, \vec{v}_1, \vec{u}_1, \ldots, \vec{v}_l, \vec{u}_l)$ in $\mathrm{comp}_\Sigma(T)$. ■

The correctness of Definition 4.2 follows immediately.

COROLLARY 4.1 (Complement of a Single Atom). *Let A be an arbitrary atom over the signature Σ and let B be a ground atom over the same signature. Then* $\mathrm{comp}_\Sigma(A)$ *according to Definition* 4.2 *represents the complement of A, i.e., B is not an instance of $A \Leftrightarrow B$ is an instance of some element in* $\mathrm{comp}_\Sigma(A)$.

Proof. Let $A = P(\vec{t})$ and let B be an arbitrary ground atom over the signature Σ with $B = Q(\vec{s})$. B is not an instance of A, iff either $P \neq Q$ or $\vec{s}$ is not an instance of $\vec{t}$. But these two cases are exactly captured by Definition 4.2. ■

Recall from the Sections 1.2 and 2.2 that we write $G_H(A)$ to denote the set of H-ground instances contained in a (possibly constrained) atom A. Then the complement representation of a single atom from Definition 4.2 can be easily extended to a representation of the complement of an ARM $\mathcal{A} = \{A_1, \ldots, A_n\}$.

COROLLARY 4.2 (Complement of an ARM). *Let $\mathcal{A} = \{A_1, \ldots, A_n\}$ be an atom set over the signature Σ and, for every $i \in \{1, \ldots, n\}$, let* $\mathrm{comp}_\Sigma(A_i)$ *be defined according to Definition* 4.2. *Then the* complement of $\mathcal{A}$ *(i.e., the set of ground atoms over Σ which are not an instance of any A_i) can be*

computed as follows:

$$\mathrm{comp}_\Sigma(\mathcal{A}) = \left[\bigcup_{B_1 \in \mathrm{comp}_\Sigma(A_1)} G_H(B_1)\right] \cap \cdots \cap \left[\bigcup_{B_n \in \mathrm{comp}_\Sigma(A_n)} G_H(B_n)\right].$$

Remark. Definition 4.2 of the complement of an atom via the complement of a term tuple improves upon a previous version of this definition in two aspects (cf. [11]): First, the inductive definition of $\mathrm{comp}_\Sigma(\vec{t})$ in Definition 4.1 can be easily implemented. Second, the number of constrained atoms necessary for representing the complement of an atom A depends linearly on the number of positions in A (rather than quadratically as in [11]). The latter fact is formally stated below.

LEMMA 4.2 (Size of the Complement Representation). *Let Σ be a signature with $|\Sigma| = c$ and let A be an atom over Σ. Moreover, let* $\mathrm{comp}_\Sigma(A)$ *be defined as in Definition* 4.2. *Then* $|\mathrm{comp}_\Sigma(A)| \leq c \times \mathrm{size}(A)$ *holds.*

Proof. Let $A = P(\vec{t})$. Then the number of atoms of the form $Q(\vec{z})$ in $\mathrm{comp}_\Sigma(A)$ with $Q \neq P$ is clearly restricted by $(c-1)$. Moreover, $\mathrm{size}(\vec{t}) = \mathrm{size}(P(\vec{t})) - 1$. Hence, it suffices to show that $|\mathrm{comp}_\Sigma(\vec{t})| \leq c \times \mathrm{size}(\vec{t})$ holds for every term tuple $\vec{t}$ over Σ. Analogously to Lemma 4.1, we proceed by induction on $\tau(\vec{t})$:

Induction Begin. If $\tau(\vec{t}) = 0$, then we distinguish two cases: If $\vec{t}$ consists of variables only, then, by Case 1 of Definition 4.1, even $|\mathrm{comp}_\Sigma(\vec{t})| \leq \mathrm{size}(\vec{t})$ holds. Otherwise, let $\vec{t} = (\vec{r}_0, a_1, \vec{r}_1, \ldots a_l, \vec{r}_l)$ with $l \geq 1$, s.t. $\vec{r}_0, \ldots, \vec{r}_l$ are vectors of variables and $a_1, \ldots, a_l$ are constant symbols. Then the number of term tuples of the form $(\vec{z}_0, w_1, \vec{z}_1, \ldots, w_l, \vec{z}_l)$ in $\mathrm{comp}_\Sigma(\vec{t})$, s.t. $w_\beta \in BT_H - \{a_\beta\}$ for exactly one $\beta \in \{1, \ldots, l\}$ is restricted by $(c-1) \times l$. Moreover, the number of elements in $\mathrm{comp}_\Sigma((\vec{r}_0, \vec{r}_1, \ldots, \vec{r}_l))$ and, therefore, also the number of tuples of the form $(\vec{u}_0, a_1, \vec{u}_1, \ldots, a_l, \vec{u}_l)$ in $\mathrm{comp}_\Sigma(\vec{t})$ are restricted by $\mathrm{size}((\vec{r}_0, \vec{r}_1, \ldots, \vec{r}_l)) = k - l$. We thus get the upper bound $|\mathrm{comp}_\Sigma(\vec{t})| \leq (c-1) \times l + (k-l) \leq c \times k = c \times \mathrm{size}(\vec{t})$.

Induction Step. Let $d \geq 0$ and suppose that $|\mathrm{comp}_\Sigma(\vec{u})| \leq c \times \mathrm{size}(\vec{u})$ holds for all term tuples $\vec{u}$ with $\tau(\vec{u}) \leq d$. Now let $\vec{t}$ be a term tuple with $\tau(\vec{t}) = d + 1$, s.t. $\vec{t}$ is of the form $\vec{t} = (\vec{r}_0, f_1(\vec{s}_1), \vec{r}_1, \ldots, f_l(\vec{s}_l), \vec{r}_l)$ with $l \geq 1$, where $\vec{r}_0, \ldots, \vec{r}_l$ are vectors of variables and $f_1(\vec{s}_1), \ldots, f_l(\vec{s}_l)$ are the nonvariable components of $\vec{t}$. Then the number of term tuples of the form $(\vec{z}_0, w_1, \vec{z}_1, \ldots w_l, \vec{z}_l)$ in $\mathrm{comp}_\Sigma(\vec{t})$, s.t. $w_\beta \in BT_H - \{f_\beta(\vec{x})\}$ for exactly one $\beta \in \{1, \ldots, l\}$ is restricted by $(c-1) \times l$.

Moreover, the number of remaining elements in $\mathrm{comp}_\Sigma(\vec{t})$ corresponds to the cardinality of $\mathrm{comp}_\Sigma((\vec{r}_0, \vec{s}_1, \vec{r}_1, \ldots, \vec{s}_l, \vec{r}_l))$. For this vector $(\vec{r}_0, \vec{s}_1, \vec{r}_1, \ldots, \vec{s}_l, \vec{r}_l)$ the conditions $\mathrm{size}((\vec{r}_0, \vec{s}_1, \vec{r}_1, \ldots, \vec{s}_l, \vec{r}_l)) = \mathrm{size}(\vec{t}) - l$ and $\tau((\vec{r}_0, \vec{s}_1, \vec{r}_1, \ldots, \vec{s}_l, \vec{r}_l)) \leq d$ hold. Hence, by the induction hypothesis, we have $|\mathrm{comp}_\Sigma((\vec{r}_0, \vec{s}_1, \vec{r}_1, \ldots, \vec{s}_l, \vec{r}_l))| \leq= c \times (\mathrm{size}(\vec{t}) - l)$. But then, $|\mathrm{comp}_\Sigma(\vec{t})| \leq (c-1) \times l + c \times (\mathrm{size}(\vec{t}) - l) \leq c \times (\mathrm{size}(\vec{t})$ holds. ■

The size of an element in $|\mathrm{comp}_\Sigma(A)|$ is clearly polynomially (in fact, even linearly) bounded w.r.t. the size of A. Moreover, by the linear bound on $|\mathrm{comp}_\Sigma(A)|$, the set $\mathrm{comp}_\Sigma(A)$ can of course be computed in polynomial time. In particular, only polynomial time is required to "guess" an arbitrary element from $\mathrm{comp}_\Sigma(A)$. The latter property will play a role in the construction of a nondeterministic clause evaluation algorithm in Section 5. Finally, recall that every atom B without constraints can be easily converted into a constrained atom by adding the trivially true constraint $\top$, i.e.: B and $[B : \top]$ have identical sets of ground instances. Hence, w.l.o.g. we may assume that all elements in $\mathrm{comp}_\Sigma(A)$ are constrained atoms.

5. coNP-MEMBERSHIP

In the case of a finite Herbrand universe, a coNP-algorithm for the CLAUSE-EVALUATION problem is easy to construct; i.e., let $C = L_1 \vee \cdots \vee L_l \vee \neg M_1 \vee \cdots \vee \neg M_m$ and let $\mathcal{A} = \{A_1, \ldots, A_n\}$ be an ARM. Then we can check by the following nondeterministic algorithm that C does not evaluate to **T** in the model $\mathcal{M}_\mathcal{A}$:

1. Guess an H-ground instance $C\vartheta$ of the clause C.

2. Check by matching that no atom $L_i\vartheta$ is an instance of any $A_k \in \mathcal{A}$ and that every atom $M_j\vartheta$ is an instance of some $A_k \in \mathcal{A}$.

However, in the presence of function symbols, there is no guarantee that the size of the "counterexample" $C\vartheta$ to be guessed in the first step is polynomially bounded. Hence, a different approach is called for. In this section, we prove the coNP-membership of the CLAUSE-EVALUATION problem by making use of the complement representation of ARMs from the previous section. By Theorem 1.1, the coNP-membership of TOTAL-COVER, MODEL-EQUIVALENCE, and ATOM-H-SUBSUMPTION will then follow immediately.

In the case of a single atom, the considerations on the complement of an ARM $\mathcal{A}$ from the previous section lead to the following truth evaluation criterion:

LEMMA 5.1 (Truth Evaluation of an Atom). *Let C be an atom over some signature Σ and let $\mathcal{A} = \{A_1, \ldots, A_n\}$ be an ARM over Σ. Furthermore, let $\mathrm{comp}_\Sigma(A_i)$ denote the complement of A_i according to Definition* 4.2. *Then the following equivalence holds*:

C evaluates to $\mathbf{F}$ in $\mathcal{M}_\mathcal{A} \Leftrightarrow$ there exist constrained atoms $B_1, \ldots, B_n$ with $B_i \in \mathrm{comp}_\Sigma(A_i)$ for every i, s.t. $G_H(C) \cap G_H(B_1) \cap \cdots \cap G_H(B_n) \neq \emptyset$.

Proof. C evaluates to $\mathbf{F}$ in $\mathcal{M}_\mathcal{A}$, iff there exists a ground instance C' of C, s.t. C' is in the complement of $\mathcal{A}$. By Corollary 4.2, this is the case iff

$$C' \in \left[\bigcup_{B_1 \in \mathrm{comp}_\Sigma(A_1)} G_H(B_1)\right] \cap \cdots \cap \left[\bigcup_{B_n \in \mathrm{comp}_\Sigma(A_n)} G_H(B_n)\right].$$

By the distributivity of $\cup$ and $\cap$, this condition holds iff there exist constrained atoms $B_1, \ldots, B_n$, with $B_i \in \mathrm{comp}_\Sigma(A_i)$ for every i, s.t. C and $G_H(B_1) \cap \cdots \cap G_H(B_n)$ have a common ground instance. ■

Now we extend the above criterion from single atoms to arbitrary clauses.

LEMMA 5.2 (Truth Evaluation of a Clause). *Let $\mathcal{A} = \{A_1, \ldots, A_n\}$ be an ARM over some signature Σ and let $C = L_1 \vee \cdots \vee L_l \vee \neg M_1 \vee \cdots \vee \neg M_m$ be a clause over Σ, where the L_i's and M_j's are unnegated atoms. Furthermore, for every $k \in \{1, \ldots, n\}$, let $\mathrm{comp}_\Sigma(A_k)$ denote the complement of A_k as in Definition* 4.2. *Then C evaluates to $\mathbf{F}$ in $\mathcal{M}_\mathcal{A}$, iff the following conditions hold.*

There exist $l \times n$ constrained atoms B_{ij} with $1 \leq i \leq l$ and $1 \leq j \leq n$, s.t. $B_{ij} \in \mathrm{comp}_\Sigma(A_j)$ for every i and j and there exist m indices $k_1, \ldots, k_m$ together with a substitution σ, s.t. $L_i\sigma \in G_H(B_{i1}) \cap \cdots \cap G_H(B_{in})$ for all $i \in \{1, \ldots, l\}$ and $M_j\sigma \in G_H(A_{k_j})$ for all $j \in \{1, \ldots, m\}$.

Proof. C evaluates to $\mathbf{F}$ in $\mathcal{M}_\mathcal{A}$, iff there exists a ground instance $C\sigma$ of C s.t. all literals $L_i\sigma$ and $\neg M_j\sigma$ evaluate to $\mathbf{F}$; i.e., every $L_i\sigma$ is in the complement of $\mathcal{A}$ and every M_j is an instance of some A_k. Analogously to the proof of Lemma 5.1, this condition holds iff there exist constrained atoms $B_{ij} \in \mathrm{comp}_\Sigma(A_j)$ with $1 \leq i \leq l$ and $1 \leq j \leq n$, s.t. $L_i\sigma \in G_H(B_{i1}) \cap \cdots \cap G_H(B_{in})$ for all $i \in \{1, \ldots, l\}$ and there exist indices $k_1, \ldots, k_m$, s.t. $M_j\sigma \in G_H(A_{k_j})$ for all $j \in \{1, \ldots, m\}$. ■

In Lemma 5.2 above we have reduced the truth evaluation of a clause C in a model represented by an ARM $\mathcal{A}$ to deciding whether certain substitutions σ on $\mathrm{Var}(C)$ exist. The following lemma provides a tool by which the existence of such a substitution can actually be checked.

LEMMA 5.3 (Simultaneous Unifications). *Let $C = L_1 \vee \cdots \vee L_l \vee \neg M_1 \vee \cdots \vee \neg M_m$ be a clause where the L_i's and M_j's are unnegated atoms. Furthermore, let $[b_{11} : \mathcal{Q}_{11}], \ldots, [b_{1n} : \mathcal{Q}_{1n}], \ldots, [b_{l1} : \mathcal{Q}_{l1}], \ldots, [b_{ln} : \mathcal{Q}_{ln}]$ be constrained atoms, and let $A_1, \ldots, A_m$ be atoms. Finally, let the clauses $C, b_{11}, \ldots, b_{1n}, \ldots, b_{l1}, \ldots, b_{ln}, A_1, \ldots, A_m$ be pairwise variable disjoint and let the simultaneous unification problem S be defined through the following set of equations*:

$$S = \{L_1 = b_{11} = \cdots = b_{1n}, \ldots, L_l = b_{l1} = \cdots = b_{ln}, M_1 = A_1, \ldots, M_m = A_m\}.$$

Then the following conditions hold:

Case 1. If S is not unifiable, then there exists no ground instance $C\lambda$ of C s.t. $L_i\lambda \in G_H([b_{i1} : \mathcal{Q}_{i1}]) \cap \cdots \cap G_H([b_{in} : \mathcal{Q}_{in}])$ for every $i \in \{1, \ldots, l\}$ and $M_j\lambda \in G_H(A_j)$ for every $j \in \{1, \ldots, m\}$.

Case 2. If S is unifiable with $\sigma = mgu(S)$, then for every ground instance $C\lambda$ of C, the following equivalence holds: $L_i\lambda \in G_H([b_{i1} : \mathcal{Q}_{i1}]) \cap \cdots \cap G_H([b_{in} : \mathcal{Q}_{in}])$ for every $i \in \{1, \ldots, l\}$ and $M_j\lambda \in G_H(A_j)$ for every $j \in \{1, \ldots, m\} \Leftrightarrow C\lambda$ is an instance of $[C\sigma : \mathcal{R}\sigma]$ with $\mathcal{R} \equiv \mathcal{Q}_{11} \wedge \cdots \wedge \mathcal{Q}_{1n} \wedge \cdots \wedge \mathcal{Q}_{l1} \wedge \cdots \wedge \mathcal{Q}_{ln}$.

Proof. If S is not unifiable, then there exists no substitution λ s.t. $L_i\lambda \in G_H(b_{i1}) \cap \cdots \cap G_H(b_{in})$ for every $i \in \{1, \ldots, l\}$ and $M_j\lambda \in G_H(A_j)$ for every $j \in \{1, \ldots, m\}$. Hence, in particular, there exists no substitution λ s.t. $L_i\lambda \in G_H([b_{i1} : \mathcal{Q}_{i1}]) \cap \cdots \cap G_H([b_{in} : \mathcal{Q}_{in}])$ and $M_j\lambda \in G_H(A_j)$ for every i and j, since the set of ground instances of an atom certainly does not increase when constraints are added. Hence, it only remains to prove the equivalence in Case 2.

"$\Rightarrow$" Let $C\lambda$ be a ground instance of C s.t. $L_i\lambda \in G_H([b_{i1} : \mathcal{Q}_{i1}]) \cap \cdots \cap G_H([b_{in} : \mathcal{Q}_{in}])$ and $M_j\lambda \in G_H(A_j)$ for every i and j. We have to show that then $C\lambda$ is an instance of $[C\sigma : \mathcal{R}\sigma]$ with $\mathcal{R} \equiv \mathcal{Q}_{11} \wedge \cdots \wedge \mathcal{Q}_{1n} \wedge \cdots \wedge \mathcal{Q}_{l1} \wedge \cdots \wedge \mathcal{Q}_{ln}$:

For every $i \in \{1, \ldots, l\}$, $L_i\lambda$ is a common ground instance of $[b_{i1} : \mathcal{Q}_{i1}]$, $\ldots$, $[b_{in} : \mathcal{Q}_{in}]$. Hence, there exist solutions φ_{i1} of $\mathcal{Q}_{i1}$, $\ldots$, φ_{in} of $\mathcal{Q}_{in}$ s.t. $L_i\lambda = b_{i1}\varphi_{i1} = \cdots = b_{in}\varphi_{in}$. By assumption, the clauses C, $b_{i1}, \ldots, b_{in}$ are variable disjoint. Now let $\varphi^{(i)} = \varphi_{i1} \cup \cdots \cup \varphi_{in}$. Then the equations $L_i(\lambda \cup \varphi^{(i)}) = L_i\lambda$, $b_{i1}(\lambda \cup \varphi^{(i)}) = b_{i1}\varphi_{i1}, \ldots, b_{in}(\lambda \cup \varphi^{(i)}) = b_{in}\varphi_{in}$ hold. Hence, $\lambda \cup \varphi^{(i)}$ is a unifier of $L_i = b_{i1} = \cdots = b_{in}$. Analogously, for every $j \in \{1, \ldots, m\}$, $M_j\lambda$ is a ground instance of A_j, i.e., $M_j\lambda = A_j\psi_j$ for some substitution ψ_j. Thus, since C and A_j have no variables in common, $\lambda \cup \psi_j$ is a unifier of $M_j = A_j$.

Remember that the clauses C, $b_{11}, \ldots, b_{1n}, \ldots, b_{l1}, \ldots, b_{ln}$, $A_1, \ldots, A_m$ are pairwise variable disjoint. Hence, we arrive at a simultaneous unifier of S by putting together all of the above unifiers λ, $\varphi^{(i)}$, and ψ_j into a single substitution, i.e., let $\mu = \lambda \cup \varphi^{(1)} \cup \cdots \cup \varphi^{(l)} \cup \psi_1 \cup \cdots \cup \psi_m$. Then μ is a unifier of S and, therefore, μ is a ground instance of $\sigma = mgu(S)$. Hence, there exists a ground substitution τ s.t. $\mu = \sigma \circ \tau$. Furthermore, $\mu|_{\mathrm{Var}(C)} = \lambda$ and, therefore, $C\lambda = C\mu$.

Moreover, $\mathrm{Var}(\mathcal{Q}_{ik}) \subseteq \mathrm{Var}(b_{ik})$ for all i and k by the definition of constrained atoms. Thus, $\mathcal{Q}_{ik}\mu \equiv \mathcal{Q}_{ik}\varphi_{ik}$ holds. Hence, $\mu = \sigma \circ \tau$ is a solution of every $\mathcal{Q}_{ik}$ and, therefore, also of $\mathcal{R} \equiv \mathcal{Q}_{11} \wedge \cdots \wedge \mathcal{Q}_{1n} \wedge \cdots \wedge \mathcal{Q}_{l1} \wedge \cdots \wedge \mathcal{Q}_{ln}$. Thus τ is a solution of $\mathcal{R}\sigma$ and $C\lambda = C\mu = C\sigma\tau$ is a ground instance of $[C\sigma : \mathcal{R}\sigma]$.

"$\Leftarrow$" Let $C\lambda$ be an instance of $[C\sigma : \mathcal{R}\sigma]$ with $\mathcal{R} \equiv \mathcal{Q}_{11} \wedge \cdots \wedge \mathcal{Q}_{1n} \wedge \cdots \wedge \mathcal{Q}_{l1} \wedge \cdots \wedge \mathcal{Q}_{ln}$. We have to show that then $L_i\lambda \in G_H([b_{i1} : \mathcal{Q}_{i1}]) \cap \cdots \cap G_H([b_{in} : \mathcal{Q}_{in}])$ for every $i \in \{1, \ldots, l\}$ and $M_j\lambda \in G_H(A_j)$ for every $j \in \{1, \ldots, m\}$ holds.

Since $C\lambda$ is a ground instance of $[C\sigma : \mathcal{R}\sigma]$, there exists a solution τ of $\mathcal{R}\sigma$ s.t. $C\lambda = C\sigma\tau$. Furthermore, by assumption, $\sigma = mgu(S)$. Hence, in particular, $L_i\sigma = b_{i1}\sigma = \cdots = b_{in}\sigma$ for every $i \in \{1, \ldots, l\}$ and $M_j\sigma = A_j\sigma$ for every $j \in \{1, \ldots, m\}$. Therefore, $L_i\lambda = L_i\sigma\tau$ is a common ground instance of $[b_{i1}\sigma : \mathcal{R}\sigma]$, $\ldots$, $[b_{in}\sigma : \mathcal{R}\sigma]$ and $M_j\lambda = M_j\sigma\tau$ is a ground instance of $A_j\sigma$ for every i and j. But every equational problem $\mathcal{Q}_{ik}$ is a conjunct in $\mathcal{R}$. Thus, $L_i\lambda = L_i\sigma\tau$ is a ground instance of $[b_{ik}\sigma : \mathcal{Q}_{ik}\sigma]$ and also of $[b_{ik} : \mathcal{Q}_{ik}]$ for every $i \in \{1, \ldots, l\}$ and $k \in \{1, \ldots, n\}$. Hence, $L_i\lambda \in G_H([b_{i1} : \mathcal{Q}_{i1}]) \cap \cdots \cap G_H([b_{in} : \mathcal{Q}_{in}])$ for every $i \in \{1, \ldots, l\}$ and $M_j\lambda \in G_H(A_j)$ for every $j \in \{1, \ldots, m\}$. ■

By combining Lemmas 5.2 and 5.3 above with Lemma 2.1 to test the nonemptiness of a set $G_H([C\sigma : \mathcal{R}\sigma])$, we are now ready to prove the coNP-membership of the CLAUSE-EVALUATION problem.

THEOREM 5.1 (coNP-Membership of CLAUSE-EVALUATION). *Let H be an arbitrary Herbrand universe. Then the CLAUSE-EVALUATION problem over H is in coNP.*

Proof. Let $C = L_1 \vee \cdots \vee L_l \vee \neg M_1 \vee \cdots \vee \neg M_m$ be a clause and let $\mathcal{A} = \{A_1, \ldots, A_n\}$ be an ARM over H. Moreover, let Σ denote the signature that contains all predicate symbols occurring in C and $\mathcal{A}$ as well as the constant symbols and function symbols from H. Then consider the following nondeterministic algorithm for testing whether C evaluates to $\mathbf{F}$ in the model represented by $\mathcal{A}$:

1. Guess $l \times n$ constrained atoms $[b_{ik} : \mathcal{Q}_{ik}]$ with $1 \leq i \leq l$ and $1 \leq k \leq n$, s.t. $[b_{ik} : \mathcal{Q}_{ik}] \in \mathrm{comp}_\Sigma(A_k)$ for every i and k.
2. Guess m indices $k_1, \ldots, k_m$ with $k_j \in \{1, \ldots, n\}$ for every j.

3. Suppose that the variables have been renamed appropriately s.t. the clauses $C, b_{11}, \ldots, b_{1n}, \ldots, b_{l1}, \ldots, b_{ln}, A_{k_1}, \ldots, A_{k_m}$ are pairwise variable disjoint. Moreover, let the simultaneous unification problem S be defined as $S = \{L_1 = b_{11} = \cdots = b_{1n}, \ldots, L_l = b_{l1} = \cdots = b_{ln}, M_1 = A_{k_1}, \ldots, M_m = A_{k_m}\}$. Then check that S is unifiable with $\sigma = mgu(S)$ and that the equational problem $\mathcal{R}\sigma$ with $\mathcal{R} \equiv \mathcal{Q}_{11} \wedge \cdots \wedge \mathcal{Q}_{1n} \wedge \cdots \wedge \mathcal{Q}_{l1} \wedge \cdots \wedge \mathcal{Q}_{ln}$ contains no trivial disequation of the form $t \neq t$.

We claim that this algorithm is correct and that its time complexity is polynomially bounded. As for the *correctness*, recall the criterion for clause evaluation from Lemma 5.2. The target of the above algorithm is to test this very criterion; namely, the purpose of Step 1 is to guess constrained atoms $B_{ik} = [b_{ik} : \mathcal{Q}_{ik}] \in \mathrm{comp}_\Sigma(A_k)$ with $1 \leq i \leq l$ and $1 \leq k \leq n$. Likewise, in Step 2 we guess m indices $k_1, \ldots, k_m$. Finally, in Step 3 the Lemmas 5.3 and 2.1 are applied to check whether an instance $C\lambda$ of C actually exists s.t. $L_i\lambda \in G_H([b_{i1} : \mathcal{Q}_{i1}]) \cap \cdots \cap G_H([b_{in} : \mathcal{Q}_{in}])$ for all $i \in \{1, \ldots, l\}$ and $M_j\lambda \in G_H(A_{k_j})$ for all $j \in \{1, \ldots, m\}$ hold.

The crucial point for the *complexity* of the above algorithm is that, by the considerations from the previous section on the complement of an ARM, the total length of the constrained atoms $[b_{ik} : \mathcal{Q}_{ik}]$ with $1 \leq i \leq l$ and $1 \leq k \leq n$, which are guessed in the first step, is polynomially bounded in the size of an input problem instance. Moreover, the use of an efficient unification algorithm guarantees that all unifications involved in the third step can be done in polynomial time (in fact, even linear time suffices; cf., e.g., [20] or [21]). Likewise, we can check in polynomial time for all disequations $u\sigma \neq v\sigma$ in the resulting equational problem $\mathcal{R}\sigma$ that the terms $u\sigma$ and $v\sigma$ are syntactically distinct. ■

In a previous version of this work, the coNP-membership of the problems TOTAL-COVER, MODEL-EQUIVALENCE, and ATOM-H-SUBSUMPTION was shown separately by a completely different method (cf. [11]). However, by Theorem 1.1, CLAUSE-EVALUATION is the hardest problem studied here and therefore the coNP-membership of the other problems is an easy consequence of Theorem 5.1. Moreover, by the coNP-hardness result from Corollary 3.1, we immediately get:

COROLLARY 5.1 (coNP-Completeness). *Let H be an arbitrary Herbrand universe with at least two elements. Then the problems TOTAL-COVER, MODEL-EQUIVALENCE, ATOM-H-SUBSUMPTION, and CLAUSE-EVALUATION over H are coNP-complete.*

6. Π_2^p-MEMBERSHIP OF CLAUSAL H-SUBSUMPTION

Analogously to the coNP-membership proof of CLAUSE-EVALUATION in the previous section, the CLAUSE-H-SUBSUMPTION problem can easily be shown to be in Π_2^p in the case of a finite Herbrand universe. To see this, we consider the following nondeterministic algorithm with NP-oracle, which checks that $\mathcal{C} \not\leq_{sH} \mathcal{D}$ holds:

1. Guess an H-ground instance $D_j\vartheta$ of some clause $D_j \in \mathcal{D}$.
2. For all clauses $C_i \in \mathcal{C}$, check by an oracle for first-order subsumption that $C_i \not\leq_s D_j\vartheta$ holds.

This algorithm clearly works in polynomial time. Furthermore, first-order subsumption is NP-complete (cf. [8, Problem LO18]). Hence, the oracle used in the above algorithm is in NP and therefore the overall algorithm is in Σ_2^p.

Again, this algorithm cannot be carried over directly to the case of an infinite Herbrand universe, since there is no guarantee that the size of the "counterexample" $D_j\vartheta$, which is guessed in the first step of this algorithm, is polynomially bounded. In order to prove the Π_2^p-membership of clausal H-subsumption also in this case, we make use of the following lemma from [7], which states that H-subsumption and ordinary subsumption coincide, if certain conditions on the term depth are fulfilled (for a proof, see [7, Lemma 6.5]).

LEMMA 6.1. *Let $\mathcal{C}$ and $\mathcal{D}$ be clause sets over some infinite Herbrand universe H and suppose that for all clauses $D \in \mathcal{D}$ and all variables x in* $\mathrm{Var}(D)$, $\tau_{\min}(x, D) > \tau(\mathcal{C})$ *holds. Then the equivalence $\mathcal{C} \leq_s \mathcal{D} \Leftrightarrow \mathcal{C} \leq_{sH} \mathcal{D}$ holds.*

Our Π_2^p-algorithm for checking $\mathcal{C} \not\leq_{sH} \mathcal{D}$ will then consist of the following steps:

1. Guess an H-instance D of some clause $D_j \in \mathcal{D}$, s.t. for all variables $x \in \mathrm{Var}(D)$, $\tau_{\min}(x, D) > \tau(\mathcal{C})$ holds.

2. For all clauses $C_i \in \mathcal{C}$, check by an oracle for first-order subsumption that $C_i \not\leq_s D$ holds.

The correctness of this algorithm follows immediately from Lemma 6.1. However, if the signature of H contains at least one function symbol of arity ≥ 2, then we have no guarantee that there exists a "counterexample" D of polynomial size. The purpose of this section is to prove that a (not necessarily ground) counterexample D of polynomial size actually does exist whenever any counterexample exists.

In [7, Theorem 6.6], a decision procedure of H- subsumption, which is based on "partial saturation" of the clause set $\mathcal{D}$ to some clause set $\mathcal{D}'$, s.t. all variables in $\mathcal{D}'$ occur at a depth greater than $\tau(\mathcal{C})$, is provided. In Definition 6.1 below we make this notion of "partial saturation" precise.

DEFINITION 6.1 (Partial Saturation). Let D be a clause over some Herbrand universe H and let $\mathrm{Var}(D) = \{x_1, \ldots, x_k\}$ denote the variables in D. Furthermore, let $BT_H^{(1)}, \ldots, BT_H^{(k)}$ denote pairwise variable disjoint variants of the linear base terms BT_H over H (cf. Section 2.1). Then an application of the *partial saturation rule* to the clause D leads to the following set of clauses:

$$\mathrm{PSat}(D) = \bigcup_{t_1 \in BT_H^{(1)}} \cdots \bigcup_{t_k \in BT_H^{(k)}} \{D[x_1 \leftarrow t_1, \ldots, x_k \leftarrow t_k]\}.$$

For a clause set $\mathcal{D}$, we define $\mathrm{PSat}(\mathcal{D}) = \bigcup_{D \in \mathcal{D}} \mathrm{PSat}(D)$. Then the result of d successive applications of the partial saturation rule to the clause set $\mathcal{D}$ will be denoted as $\mathcal{D}^d$; i.e., $\mathcal{D}^0 = \mathcal{D}$ and $\mathcal{D}^i = \mathrm{PSat}(\mathcal{D}^{i-1})$ for $i \geq 1$.

The following properties of the set $\mathcal{D}^d$, which results from d successive applications of the partial saturation rule, form the starting point of our construction of a counterexample of polynomial size:

LEMMA 6.2 (Properties of $\mathcal{D}^d$). *Let $\mathcal{C}$ and $\mathcal{D}$ be clause sets over some Herbrand universe H. Furthermore, let $d = \max(\{\tau(\mathcal{C}), \tau(\mathcal{D})\}) + 1$ and let the set $\mathcal{D}^d$ be defined according to Definition* 6.1. *Then $\mathcal{D}^d$ has the following properties*:

1. *$\mathcal{D}^d =_{sH} \mathcal{D}$, i.e., $\mathcal{D}^d$ and $\mathcal{D}$ represent the same set of ground instances.*
2. *$\tau(\mathcal{D}^d) < 2d$.*
3. *For every $D \in \mathcal{D}^d$ and every variable $x \in \mathrm{Var}(D)$, $\tau_{\min}(x, D) \geq d$.*

Proof. We first consider a single application of the partial saturation rule.

1. $\mathcal{D}^1 =_{sH} \mathcal{D}$: Remember from Section 2.1 that the linear base terms BT_H are defined in such a way that they H-subsume the whole Herbrand universe H, i.e., $BT_H =_{sH} H$. Hence, for an arbitrary clause D with variables $\mathrm{Var}(D) = \{x_1, \ldots, x_k\}$, the following clause sets have the same set of ground clauses:

$$\begin{aligned}
\{D\} &=_{sH} \bigcup_{t_1 \in BT_H^{(1)}} \{D[x_1 \leftarrow t_1]\} =_{sH} \\
&=_{sH} \bigcup_{t_1 \in BT_H^{(1)}} \bigcup_{t_2 \in BT_H^{(2)}} \{D[x_1 \leftarrow t_1, x_2 \leftarrow t_2]\} =_{sH} \cdots =_{sH} \\
&=_{sH} \bigcup_{t_1 \in BT_H^{(1)}} \bigcup_{t_2 \in BT_H^{(2)}} \cdots \bigcup_{t_k \in BT_H^{(k)}} \{D[x_1 \leftarrow t_1, \ldots, x_k \leftarrow t_k]\} = \mathrm{PSat}(D).
\end{aligned}$$

2. $\tau(\mathcal{D}^1) \leq \tau(\mathcal{D}) + 1$: For an arbitrary clause D with variables $\mathrm{Var}(D) = \{x_1, \ldots, x_k\}$ and for any k-tuple of terms $(t_1, \ldots, t_k)$, the following inequality holds: $\tau(D[x_1 \leftarrow t_1, \ldots, x_k \leftarrow t_k]) \leq \tau(D) + \max(\{\tau(t_1), \ldots, \tau(t_k)\})$. Furthermore, the term depth of the linear base terms is either 0 or 1. Hence, if every variable $x_i \in \mathrm{Var}(D)$ is instantiated to a linear base term t_i, we get the following

inequality: $\tau(D[x_1 \leftarrow t_1, \ldots, x_k \leftarrow t_k]) \leq \tau(D) + 1$. By applying this upper bound to every clause $D \in \mathcal{D}$, we arrive at the desired inequality $\tau(\mathcal{D}^1) \leq \tau(\mathcal{D}) + 1$.

3. Let e denote a lower bound on the depth of variable occurrences in $\mathcal{D}$, i.e., for every $D \in \mathcal{D}$ and every $x_i \in \mathrm{Var}(D)$, $\tau_{\min}(x_i, D) \geq e$ holds. We claim that then $e+1$ is a lower bound on the depth of variable occurrences in $\mathcal{D}^1$: By the definition of the partial saturation rule, every variable x_i in a clause $D \in \mathcal{D}$ is either replaced by a constant or by a term of the form $f(\vec{v})$, where $\vec{v}$ is a vector of variables. Thus $\tau_{\min}(v_j, D') \geq e + 1$ for every clause $D' \in \mathrm{PSat}(D)$ and every variable v_j in D'.

For the initial set $\mathcal{D}^0 = \mathcal{D}$, the conditions $\tau(\mathcal{D}^0) < d$ and $\tau_{\min}(x, D) \geq 0$ hold for every $D \in \mathcal{D}^0$ and every $x \in \mathrm{Var}(D)$. Hence, Lemma 6.2 follows by an easy induction argument on the number of partial saturation rule applications. ■

Remember from Definition 1.1 that $\mathcal{C} \leq_s D$ holds, iff there exists a clause $C \in \mathcal{C}$ and a substitution ϑ s.t. $C\vartheta \subseteq D$. In this definition of subsumption, clauses are considered as sets of literals. In this section, it is more convenient to consider the order of literals in a clause as fixed. Moreover, the predicate symbol of a literal together with its sign is considered as a single symbol (i.e., the "literal symbol"). Then the notion of positions can be extended in a natural way from atoms to clauses, namely, the position $p = p_1 p_2 \cdots p_k$ with $p \neq \varepsilon$ in some clause C denotes the position $p_2 \cdots p_k$ of the p_1th literal of C. Furthermore, a clause $B_1 \vee \cdots \vee B_n$ is an instance of another clause $A_1 \vee \cdots \vee A_n$, iff there exists a substitution ϑ s.t. $A_i\vartheta = B_i$ for every $i \in \{1, \ldots, n\}$. Then the following criterion for subsumption of clauses $C = L_1 \vee \cdots \vee L_l$ and $D = M_1 \vee \cdots \vee M_m$ clearly holds: $C \leq_s D$, iff there exists an l-tuple of indices $(k_1, \ldots, k_l) \in \{1, \ldots, m\}^l$ s.t. the clause $M_{k_1} \vee \cdots \vee M_{k_l}$ is an instance of $C = L_1 \vee \cdots \vee L_l$.

On the other hand, if some clause $D' = M_{k_1} \vee \cdots \vee M_{k_l}$ is *not* an instance of $C = L_1 \vee \cdots \vee L_l$, then this is detected by a simple matching algorithm in the following way: Either there is a nonvariable position p in C, s.t. $[C \mid p]$ and $[D \mid p]$ have different leading symbols, or there is a variable x with two distinct occurrences p and q in C, s.t. the subterms in D at the positions p and q are not identical. In other words, only a very limited number of positions in D' is responsible, if D' is not an instance of C. Likewise, if $\mathcal{C} \not\leq_s D$, then this is also due to a restricted number of positions in D. This observation is formalized in the following definition and lemma of "witnesses" for the nonsubsumption of a clause D by a clause set $\mathcal{C}$.

DEFINITION 6.2 (Witness Branches). Let $\mathcal{C} = \{C_1, \ldots, C_n\}$ be a clause set and D be a clause with $C_i = L_{i1} \vee \cdots \vee L_{il_i}$ for $i \in \{1, \ldots, n\}$ and $D = M_1 \vee \cdots \vee M_m$. Furthermore, assume that $\mathcal{C} \not\leq_s D$ holds. Then we define the set Wit of *witness branches* in the following way:

$$\mathrm{Wit} = \bigcup_{i=1}^{n} \mathrm{SWit}_i \cup \bigcup_{i=1}^{n} \mathrm{PWit}_i,$$

where the sets SWit_i and PWit_i are defined as follows.

SWit_i (=Witnesses of Single Literals). Let L_{ij} with $j \in \{1, \ldots, l_i\}$ be a literal from C_i and let M_k with $k \in \{1, \ldots, m\}$ be a literal from D. If M_k is not an instance of L_{ij}, then we distinguish the following two cases:

- *Case 1.* There exists a nonvariable position p in L_{ij}, s.t. $[L_{ij} \mid p]$ and $[M_k \mid p]$ have different leading symbols. Moreover, let p be the minimal position (w.r.t. the lexicographical ordering on positions mentioned in Section 2.1) with this property and let π be the minimal branch in D s.t. the position p in M_k is on this branch. Then SWit_i contains the branch π.
- *Case 2.* There exist two distinct positions p and q in L_{ij}, s.t. $[L_{ij} \mid p] = [L_{ij} \mid q] = x$ for some variable x and $[M_k \mid p] \neq [M_k \mid q]$. Hence, in particular, there exists a position r in the terms $[M_k \mid p]$ and $[M_k \mid q]$ s.t. the terms $[M_k \mid p \circ r]$ and $[M_k \mid q \circ r]$ have different leading symbols. Moreover, let the positions p, q, and r be minimal with these properties and let π_1 and π_2 in D be the minimal branches, s.t. the position $p \circ r$ in M_k is situated on π_1 and $q \circ r$ is on π_2, respectively. Then SWit_i contains the two branches π_1 and π_2.

PWit$_i$ (=Witnesses of Pairs of Literals). If $l_i = 1$ (i.e., C_i is a unit clause), then we set PWit$_i = \emptyset$. Otherwise we consider every pair of literals $L_{ij_1} \vee L_{ij_2}$ from C_i with $1 \leq j_1 < j_2 \leq l_i$ and every pair of literals $M_{k_1} \vee M_{k_2}$ from D for arbitrary $(k_1, k_2) \in \{1, \ldots, m\}^2$. Suppose that there exists a position p in L_{ij_1} and a position q in L_{ij_2} s.t. $[L_{ij_1} \mid p] = [L_{ij_2} \mid q] = x$ for some variable x and $[M_{k_1} \mid p] \neq [M_{k_2} \mid q]$. Then, there exists a position r in the terms $[M_{k_1} \mid p]$ and $[M_{k_2} \mid q]$ s.t. $[M_{k_1} \mid p \circ r]$ and $[M_{k_2} \mid q \circ r]$ have different leading symbols. Moreover, let the positions p, q and r be minimal with these properties and let π_1 and π_2 in D be the minimal branches, s.t. the position $p \circ r$ in M_{k_1} is situated on π_1 and π_2 goes through the position $q \circ r$ in M_{k_2}, respectively. Then P*Wit*$_i$ contains the two branches π_1 and π_2.

The requirement of choosing the minimal positions with certain properties and the minimal paths through these positions in the above definition is not really necessary. However, the proof of Lemma 6.3, where we claim that the set of witnesses Wit provides a sufficient criterion for the nonsubsumption of an arbitrary clause E by the clause set $\mathcal{C}$, will be easier when the positions and paths involved are unique.

LEMMA 6.3 (Nonsubsumption Criterion Based on Wit). *Let $\mathcal{C} = \{C_1, \ldots, C_n\}$ be a clause set with $C_i = L_{i1} \vee \cdots \vee L_{il_i}$ for $i \in \{1, \ldots, n\}$ and let $D = M_1 \vee \cdots \vee M_m$ be a clause s.t. $\mathcal{C} \not\leq_s D$. Furthermore, let the set of witnesses* Wit *be defined according to Definition* 6.2. *If $E = N_1 \vee \cdots \vee N_m$ is another clause s.t. D and E coincide on all branches in* Wit, *then $\mathcal{C} \not\leq_s E$ also holds.*

Proof. Suppose that $\mathcal{C} \not\leq_s D$ holds. Then, by the criterion for nonsubsumption of clauses mentioned earlier in this section, there exists no $C_i \in \mathcal{C}$ and no l_i-tuple of indices $(k_1, \ldots, k_{l_i}) \in \{1, \ldots, m\}^{l_i}$, s.t. the clause $M_{k_1} \vee \cdots \vee M_{k_{l_i}}$ is an instance of $C_i = L_{i1} \vee \cdots \vee L_{il_i}$. Then we have to prove that the same condition also holds for E; i.e., for all $C_i \in \mathcal{C}$ and all $(k_1, \ldots, k_{l_i}) \in \{1, \ldots, m\}^{l_i}$ we have to show that $E' = N_{k_1} \vee \cdots \vee N_{k_{l_i}}$ is not an instance of C_i.

By assumption, $D' = M_{k_1} \vee \cdots \vee M_{k_{l_i}}$ is not an instance of C_i. Then we distinguish the following cases (which correspond to the cases in the definition of Wit).

Case 1. There exists a nonvariable position $p = p_1 \cdots p_\alpha$ in C_i, s.t. $[C_i \mid p]$ and $[D' \mid p]$ have different leading symbols. Without loss of genarality we assume that p is minimal with this property. Then $p' = p_2 \cdots p_\alpha$ is a nonvariable position, s.t. $[M_{k_{p_1}} \mid p']$ and $[L_{ip_1} \mid p']$ have different leading symbols. Thus, $M_{k_{p_1}}$ is not an instance of L_{ip_1} and, by Definition 6.2, SWit$_i$ contains a branch π which goes through p. But E and D coincide on this branch. Hence, $N_{k_{p_1}}$ is not an instance of L_{ip_1} either and, therefore, $E' = N_{k_1} \vee \cdots \vee N_{k_{l_i}}$ is not an instance of $C_i = L_{i1} \vee \cdots \vee L_{il_i}$.

Case 2. There exists a variable x with two occurrences $p = p_1 \cdots p_\alpha$ and $q = q_1 \cdots q_\beta$ in C_i s.t. $[D' \mid p] \neq [D' \mid q]$. Hence, in particular, there exists a position r in the terms $[D' \mid p]$ and $[D' \mid q]$ s.t. $[D' \mid p \circ r]$ and $[D' \mid q \circ r]$ have different leading symbols. Again we assume w.l.o.g. that the positions p, q, and r are minimal with these properties. Then we have to distinguish the following two subcases:

Case 2.1. $p_1 = q_1$; i.e., both occurrences of x are in the same literal L_{ip_1}. Then, in particular, the literal $M_{k_{p_1}}$ is not an instance of L_{ip_1} and, by Definition 6.2, SWit$_i$ contains a pair of branches (π_1, π_2) which go through the positions $p \circ r$ and $q \circ r$, respectively. But E and D coincide on these two branches. Hence, $N_{k_{p_1}}$ is not an instance of L_{ip_1} and, therefore, $E' = N_{k_1} \vee \cdots \vee N_{k_{l_i}}$ is not an instance of $C_i = L_{i1} \vee \cdots \vee L_{il_i}$.

Case 2.2. $p_1 \neq q_1$; i.e., the two occurrences of x are in different literals L_{ip_1} and L_{iq_1}. Then, in particular, the two-literal clause $M_{k_{p_1}} \vee M_{k_{q_1}}$ is not an instance of $L_{ip_1} \vee L_{iq_1}$ and, by Definition 6.2, PWit$_i$ contains a pair of branches (π_1, π_2) which go through the positions $p \circ r$ and $q \circ r$, respectively. Again, since E and D coincide on these two branches, $N_{k_{p_1}} \vee N_{k_{q_1}}$ is not an instance of $L_{ip_1} \vee L_{iq_1}$ and, therefore, $E' = N_{k_1} \vee \cdots \vee N_{k_{l_i}}$ is not an instance of $C_i = L_{i1} \vee \cdots \vee L_{il_i}$. ■

Our construction of a counterexample D^* of polynomial size w.r.t. the input problem instance "$\mathcal{C} \leq_{sH} \mathcal{D}$" will be based on the following idea: We start off with a clause $D \in \mathcal{D}^d$ which is not subsumed by $\mathcal{C}$ and "reduce" D to a smaller clause D^* by pruning "irrelevant" subtrees from D. If we leave all witness branches in D unchanged then, by Lemma 6.3, we can be sure that the "reduced" clause is not subsumed by $\mathcal{C}$ either. Note that $D \in \mathcal{D}^d$ is an instance of some clause D_j from the input set $\mathcal{D}$. In order to construct a counterexample D^* for the H-subsumption problem "$\mathcal{C} \leq_{sH} \mathcal{D}$" we have to make

sure that all H-ground instances of D^* are instances of clauses in $\mathcal{D}$. To this end, we shall construct D^* in such a way that it is still an instance of D_j. But then we have to take multiple variable occurrences in D_j into account. In order to make sure that all occurrences of a variable x in D_j are instantiated to the same term in D^*, we introduce the notion of *similbranches*.

DEFINITION 6.3 (Similbranches). Let D_j be a clause over some Herbrand universe H and let D be an instance of D_j. Furthermore, let π be a branch in D. Then we define the set $\mathrm{Sim}(\pi)$ of similbranches of π as follows:

Suppose that the corresponding branch in D_j goes through a variable, i.e., $\pi = p_1 \cdots p_n$ and there exists a position $p' = p_1 \cdots p_k$ with $k \leq n$, s.t. $[D_j \mid p'] = x$ for some variable x°, then we define $\mathrm{Sim}(\pi) = \bigcup_{q \in Q}\{q \circ p_{k+1} \cdots p_n\}$, where $Q = \{q : [D_j \mid q] = x\}$ denotes the set of all occurrences of x in D_j.

If the branch in D_j corresponding to π does not go through a variable, then we set $\mathrm{Sim}(\pi) = \{\pi\}$.

Finally, for a set Π of branches in D, we define $\mathrm{Sim}(\Pi) = \bigcup_{\pi \in \Pi} \mathrm{Sim}(\pi)$.

Note that every *similbranch* $\rho \in \mathrm{Sim}(\pi)$ of a branch $\pi = p_1 \cdots p_n$ is actually a *branch* in D. In order to see this, let $Q = \{q : [D_j \mid q] = x\}$ and let $p' \in Q$ be a position in D_j with $p' = p_1 \cdots p_k$ for some $k \leq n$. By assumption, $D = D_j\vartheta$ for some substitution ϑ and, therefore, $[D \mid q] = x\vartheta$ for every $q \in Q$. In particular, $[D \mid p'] = x\vartheta$. But then $p_{k+1} \cdots p_n$ is a branch in the term $x\vartheta$ and, therefore, $q \circ p_{k+1} \cdots p_n$ is a branch in D for every $q \in Q$.

In our construction of a polynomial size counterexample D^* from an arbitrary counterexample $D \in \mathcal{D}^d$, we only have to take care of those positions in D, which guarantee that, on the one hand, D^* is not subsumed by $\mathcal{C}$ and, on the other hand, that D^* is still an instance of some $D_j \in \mathcal{D}$. This idea is made precise in the notion of "relevant" positions defined below. Moreover, it will turn out that there are only polynomially many such positions.

DEFINITION 6.4 (Relevant Positions). Let $\mathcal{C}$ and $\mathcal{D}$ be clause sets and let $D \in \mathcal{D}^d$ be an instance of some clause $D_j \in \mathcal{D}$ with $\mathcal{C} \not\leq_s D$. Furthermore, let Wit denote the set of witnesses that D is not subsumed by $\mathcal{C}$ according to Definition 6.2. Then the set $\mathrm{Rel}(D)$ of *relevant positions* in D is defined as follows:

1. All positions of D_j are in $\mathrm{Rel}(D)$.
2. All positions in all similbranches of Wit are in $\mathrm{Rel}(D)$.

LEMMA 6.4 (Number of Relevant Positions). *Let $\mathcal{C} = \{C_1, \ldots, C_n\}$ and $\mathcal{D} = \{D_1, \ldots, D_m\}$ be clause sets with $d = \max(\{\tau(\mathcal{C}), \tau(\mathcal{D})\}) + 1$. Moreover, let L denote an upper bound on the number of literals in the clause D and in the clauses $C_i \in \mathcal{C}$. Finally, suppose that $D \in \mathcal{D}^d$ is an instance of some clause $D_j \in \mathcal{D}$ with $\mathcal{C} \not\leq_s D$ and let* Rel(D) *be defined as in Definition* 6.4. *Then the number of positions in* Rel(D) *is restricted in the following way*: $|\mathrm{Rel}(D)| \leq \mathrm{size}(D_j) \times [1 + n \times (2L^2 + L^4) \times 2d]$.

Proof. The number of positions in D_j, by definition, corresponds to $\mathrm{size}(D_j)$. It remains to prove that the number of positions in the similbranches of Wit is restricted by $\mathrm{size}(D_j) \times n \times (2L^2 + L^4) \times 2d$.

The maximum number of positions along a branch π in D is restricted by the term depth of D which, by Lemma 6.2, is strictly smaller than $2d$. Now suppose that π is an arbitrary branch in D, s.t. the corresponding branch in D_j goes through a variable x. The maximum number of occurrences of x in D_j is clearly restricted by $\mathrm{size}(D_j)$. Hence, also $|\mathrm{Sim}(\pi)| \leq \mathrm{size}(D_j)$ holds. But then the number of positions in similbranches of witness branches is restricted by $\mathrm{size}(D_j) \times |\mathrm{Wit}| \times 2d$.

Now the only part missing in our proof is an appropriate upper bound on $|\mathrm{Wit}|$: Let $C_i = L_{i1} \vee \cdots \vee L_{il_i}$ for $i \in \{1, \ldots, n\}$ and let $D = M_1 \vee \cdots \vee M_m$. Then $m \leq L$ and $l_i \leq L$ for all $i \in \{1, \ldots, n\}$ hold by assumption. Moreover, the following inequalities hold by the construction of Wit:

$$|\mathrm{Wit}| \leq \sum_{i=1}^{n} |\mathrm{SWit}_i| + \sum_{i=1}^{n} |\mathrm{PWit}_i|,$$

with

$$|\mathrm{SWit}_i| \leq 2 \times L^2$$

and

$$|\mathrm{PWit}_i| \leq 2 \times \left|\left\{L_{ij_1} \vee L_{ij_2} \mid 1 \leq j_1 < j_2 \leq l_i\right\}\right| \left|\left\{M_{k_1} \vee M_{k_2} \mid k_1 \leq m \text{ and } k_2 \leq m\right\}\right|$$
$$\leq 2 \times \binom{L}{2} \times L^2 \leq L^4.$$

We thus get the desired bound $|\mathrm{Wit}| \leq (n \times 2L^2) + (n \times L^4) = n \times (2L^2 + L^4)$. ■

In order to prune all subtrees with no relevant positions from the clause D, we define the following *cutting rule*.

DEFINITION 6.5 (Cutting Rule). Let $\mathcal{C}$ and $\mathcal{D}$ be clause sets over some Herbrand universe H and let "a" denote a constant symbol in H. Furthermore, let $D \in \mathcal{D}^d$ be an H-instance of some clause $D_j \in \mathcal{D}$ s.t. D is not subsumed by $\mathcal{C}$, and let the set of *relevant positions* $\mathrm{Rel}(D)$ be defined according to Definition 6.4. Then we define the cutting rule as follows.

Let p be a relevant position in D, s.t. $[D \mid p] = F(t_1, \ldots, t_i, \ldots, t_\alpha)$ holds, where F is a function symbol or a (possibly negated) predicate symbol and $\alpha \geq 1$ denotes the arity of F. Moreover, suppose that no position in t_i is relevant. Then the subtree $F(t_1, \ldots, t_i, \ldots, t_\alpha)$ may be replaced by $F(t_1, \ldots, t_{i-1}, a, t_{i+1}, \ldots, t_\alpha)$.

The following lemma shows that, in order to check whether a subtree in D contains no relevant positions, it suffices to inspect its root node.

LEMMA 6.5 (Irrelevant Positions in D). *Let p be an irrelevant position in D, i.e., $p \notin \mathrm{Rel}(D)$. Then every position q below p is also irrelevant; i.e., for every position $r \neq \varepsilon$, $p \circ r \notin \mathrm{Rel}(D)$.*

Proof. Let p and $q = p \circ r$ be positions in D and suppose that $q \in \mathrm{Rel}(D)$. We have to show that then also $p \in \mathrm{Rel}(D)$. Note that, by the definition of $\mathrm{Rel}(D)$, q is either a position in D_j or a position on a similbranch π of some witness branch. In the former case every subposition of q is also a position in D_j, and in the latter case every subposition of q is also a position on π. Hence, p is relevant, too. ■

In the following lemmas, we shall prove several properties of the clause D^*, which results from exhaustively applying the cutting rule to an arbitrary counterexample $D \in \mathcal{D}^d$. In Lemma 6.9, these lemmas will then be used to show that the clause D^* is the desired counterexample of polynomial size.

LEMMA 6.6 (Positions of D^*). *Let $D \in \mathcal{D}^d$ be an H-instance of some clause $D_j \in \mathcal{D}$ with $\mathcal{C} \not\leq_s D$ and let D^* be the clause resulting from exhaustively applying the cutting rule from Definition* 6.5 *to the clause D. Then the following conditions on the positions of D^* hold*:

1. *Every position p of D^* is also a position of D.*
2. *If $p \in \mathrm{Rel}(D)$, then p is also a position of D^* and, furthermore, $[D^* \mid p]$ has the same leading symbol as $[D \mid p]$.*
3. *If p is a position in D^* and $p \notin \mathrm{Rel}(D)$, then $[D^* \mid p] = a$.*

Proof. An application of the cutting rule can never introduce a new position, since its only effect is to prune the subtree at some position p and to replace the term at position p by a. Hence, all positions of D^* are also contained in D.

If p is a relevant position in D, then the cutting rule must not be applied to the position p or any position above p. Hence, neither can a relevant position p ever be deleted nor can the leading symbol of an expression at position p be altered by an application of the cutting rule. But then, on the one hand, position p still exists in D^* and, on the other hand, $[D^* \mid p]$ and $[D \mid p]$ have the same leading symbol.

If $p = p_1 \cdots p_k$ is an irrelevant position of D^* then, by Lemma 6.5, also all positions below p are irrelevant. Now suppose that $[D^* \mid p] \neq a$ and $[D^* \mid p'] = F(t_1, \ldots, t_{p_k}, \ldots, t_\alpha)$ for the position $p' = p_1 \cdots p_{k-1}$. Then the cutting rule may be applied to replace $F(t_1, \ldots, t_{p_k}, \ldots, t_\alpha)$ by $F(t_1, \ldots, a, \ldots, t_\alpha)$. But, by assumption, no more cutting rule application to D^* is possible. Hence, $[D^* \mid p] = t_{p_k} = a$. ■

LEMMA 6.7 (Further Properties of D^*). *Let $D \in \mathcal{D}^d$ be an H-instance of some clause $D_j \in \mathcal{D}$ with $\mathcal{C} \not\leq_s D$ and let D^* be the clause resulting from exhaustively applying the cutting rule from Definition* 6.5 *to the clause D. Then D^* has the following properties*:

1. *D^* is also an H-instance of D_j.*
2. *$\mathcal{C} \not\leq_s D^*$.*
3. *For every variable $x \in \mathrm{Var}(D^*)$, $\tau_{\min}(x, D^*) \geq d$.*

Proof. For the proof of Property 1, we suppose on the contrary that D^* is *not* an H-instance of D_j. Then we derive a contradiction for the following two possibilities.

Case 1. There exists a nonvariable position p in D_j, s.t. $[D_j \mid p]$ and $[D^* \mid p]$ have different leading symbols. In particular, p is a position in D_j and thus $p \in \mathrm{Rel}(D)$. Hence, by Lemma 6.6, $[D \mid p]$ and $[D^* \mid p]$ have the same leading symbol. But then D is not an instance of D_j either.

Case 2. There exist positions p and q in D_j, s.t. $[D_j \mid p] = [D_j \mid q] = x$ for some variable x and $[D^* \mid p] \neq [D^* \mid q]$; i.e., there exists a position r in $[D^* \mid p]$ and $[D^* \mid q]$, s.t. $[D^* \mid p \circ r]$ and $[D^* \mid q \circ r]$ have different leading symbols. Note that r cannot be the empty position. For suppose on the contrary that $r = \varepsilon$. Then $p \circ r = p$ and $q \circ r = q$ are positions in D_j and, therefore, $p \circ r \in \mathrm{Rel}(D)$ and $q \circ r \in \mathrm{Rel}(D)$. But then, by Lemma 6.6, $[D \mid p \circ r]$ and $[D \mid q \circ r]$ have the same leading symbols as $[D^* \mid p \circ r]$ and $[D^* \mid q \circ r]$, respectively, and thus D is not an instance of D_j either. So it only remains to consider the case that $r \neq \varepsilon$.

By assumption, the terms at the positions $p \circ r$ and $q \circ r$ in D^* have different leading symbols. Hence, at least one of these two positions must be relevant, since otherwise, by Lemma 6.6, we would have the term a on both positions. Without loss of generality we assume that $p \circ r \in \mathrm{Rel}(D)$. Note that $p \circ r$ is not a position in D_j, since p is a variable position in D_j (i.e., p is a leaf node in D_j) and we only consider the case that $r \neq \varepsilon$. But then, by the definition of relevant positions, $p \circ r$ is on a similbranch of some witness branch; i.e., there exists a position p' in D_j, s.t. $[D_j \mid p'] = x$ and $p' \circ r$ is a position on a witness branch $\pi \in \mathrm{Wit}$. If $p' = q$, then $q \circ r$ is on the witness branch π, and if $p' \neq q$, then $q \circ r$ is on a similbranch w.r.t. π. In either case, $q \circ r$ is also a relevant position. Hence, by Lemma 6.6, the leading symbols of the terms in D and D^* coincide on the positions $p \circ r$ and $q \circ r$. But then, $[D \mid p \circ r]$ and $[D \mid q \circ r]$ have different leading symbols. Thus, D is not an instance of D_j.

For Property 2 note that, by Lemma 6.6, D^* coincides with D on all relevant positions and hence, in particular, on all witness branches. But then, by Lemma 6.3, D^* is not subsumed by $\mathcal{C}$. Finally, for Property 3, remember from Lemma 6.6 that at all irrelevant positions in D^* we have the term a. Hence, every variable x in D^* occurs at a relevant position p. But then, by Lemma 6.6, D^* and D coincide on p. Thus, $\tau_{\min}(x, D^*) \geq \tau_{\min}(x, D) \geq d$ holds for every $x \in \mathrm{Var}(D^*)$. ■

LEMMA 6.8 (Size of D^*). *Let $\mathcal{C} = \{C_1, \ldots, C_n\}$ and $\mathcal{D} = \{D_1, \ldots, D_m\}$ be clause sets over some signature Σ and let H be the corresponding Herbrand universe. Let $D \in \mathcal{D}^d$ be an H-instance of some clause $D_j \in \mathcal{D}$ with $\mathcal{C} \not\leq_s D$ and let D^* be the clause resulting from exhaustively applying the cutting rule from Definition* 6.5 *to the clause D. Finally, let c denote the maximum arity of the symbols in Σ. Then* size(D^*) *has the following upper bound*:

$$\mathrm{size}(D^*) \leq (c+1) \times \mathrm{size}(D_j) \times [1 + n \times (2L^2 + L^4) \times 2d].$$

Proof. Recall that we have defined the size of a clause as the number of positions (or, equivalently, as the number of nodes in the tree representation). By Lemma 6.6, all nodes corresponding to irrelevant positions in D^* are labelled with the constant symbol a, i.e., all irrelevant positions in D^* correspond to leaf nodes and, therefore, all internal nodes of D^* are relevant. Hence, we get the upper bound $\mathrm{size}(D^*) \leq |\mathrm{Rel}(D)| + |\{\text{leaf nodes of } D^*\}|$.

The degree of an internal node p in D^* (i.e., the number of child nodes of such a node) corresponds to the arity of the function symbol or the (possibly negated) predicate symbol labelling p. Hence, the degree of the nodes in the tree representation of D^* is restricted by the maximum arity c of the symbols in Σ. Furthermore, the number of leaf nodes is restricted by the number of nodes immediately above the

leaf nodes times their maximum degree. Moreover, all parents of leaf nodes are internal nodes. Hence, we get the following upper bound: $|\{\text{leaf nodes of } D^*\}| \leq c \times |\{\text{internal nodes of } D^*\}| \leq c \times |\text{Rel}(D)|$.

By putting these pieces together, we have $\text{size}(D^*) \leq (c+1) \times |\text{Rel}(D)|$. Then, by putting this together with the relation $|\text{Rel}(D)| \leq \text{size}(D_j) \times [1 + n \times (2L^2 + L^4) \times 2d]$ from Lemma 6.4, we get the desired upper bound on the size of D^*. ■

In Lemmas 6.7 and 6.8 we have basically shown that D^* is an appropriate counterexample. In particular, we have established a polynomial bound on the size of D^*. Hence, it is now easy to prove that a polynomial-size counterexample exists whenever any counterexample exists.

LEMMA 6.9 (Existence of a Counterexample of Polynomial Size). *Let $\mathcal{C}$ and $\mathcal{D}$ be clause sets over some signature Σ and let H be the corresponding Herbrand universe. Furthermore, let $d = \max(\{\tau(\mathcal{C}), \tau(\mathcal{D})\}) + 1$ and let L denote an upper bound on the number of literals in the clauses $C_i \in \mathcal{C}$ and in the clauses $D_j \in \mathcal{D}$. Finally, let c denote the maximum arity of the symbols in Σ. Then the following equivalence holds*:
$\mathcal{C} \not\leq_{sH} \mathcal{D}$, *iff there exists an H-instance D^* of some $D_j \in \mathcal{D}$, s.t. the following properties hold*:

1. $\text{size}(D^*) \leq (c+1) \times \text{size}(D_j) \times [1 + |\mathcal{C}| \times (2L^2 + L^4) \times 2d]$;
2. *for every variable* $x \in \text{Var}(D^*)$, $\tau_{\min}(x, D^*) \geq d$;
3. $\mathcal{C} \not\leq_s D^*$.

Proof. We prove both directions of the equivalence separately.

"$\Rightarrow$" If $\mathcal{C} \not\leq_{sH} \mathcal{D}$, then also $\mathcal{C} \not\leq_{sH} \mathcal{D}^d$, since $\mathcal{D}^d =_{sH} \mathcal{D}$ holds by Lemma 6.2. But then $\mathcal{C} \not\leq_s \mathcal{D}^d$ holds as well; i.e., there exists a clause $D \in \mathcal{D}^d$, s.t. $\mathcal{C} \not\leq_s D$. From D we can construct D^* by exhaustive application of the cutting rule from Definition 6.5. Then, by Lemmas 6.7 and 6.8, D^* has the desired properties.

"$\Leftarrow$" Let D^* be an instance of some clause $D_j \in \mathcal{D}$ with the above properties. Then, in particular, the conditions $\mathcal{C} \not\leq_s D^*$ and $\tau_{\min}(x, D^*) \geq d$ for every variable $x \in \text{Var}(D^*)$ hold. Hence, by Lemma 6.1, also $\mathcal{C} \not\leq_{sH} D^*$ holds; i.e., there exists an H-ground instance D' of D^*, s.t. D' is not subsumed by $\mathcal{C}$. But D' is also an H-ground instance of $D_j \in \mathcal{D}$ and, therefore, $\mathcal{C} \not\leq_{sH} D_j$ holds as well. ■

The above criterion for testing that $\mathcal{C}$ does not H-subsume $\mathcal{D}$ will now be put to work in the Π_2^p-membership proof of the clausal H-subsumption problem:

THEOREM 6.1 (Π_2^p-Membership of CLAUSE-H-SUBSUMPTION). *Let H be an arbitrary Herbrand universe. Then the CLAUSE-H-SUBSUMPTION problem over H is in Π_2^p.*

Proof. The Π_2^p-membership in the case of a finite Herbrand universe is clear. Hence, we only have to consider the case of an infinite Herbrand universe H. Let $\mathcal{C}$ and $\mathcal{D}$ be clause sets over H. Moreover, let Σ denote the signature that contains all predicate symbols occurring in $\mathcal{C}$ and $\mathcal{A}$ as well as the constant symbols and function symbols from H. Then the following nondeterministic algorithm with first-order subsumption oracle checks in polynomial time that $\mathcal{C} \not\leq_{sH} \mathcal{D}$ holds.

1. Guess an H-instance D^* of some clause $D_j \in \mathcal{D}$ with $\tau_{\min}(x, D^*) \geq d$ for every $x \in \text{Var}(D^*)$ and $\text{size}(D^*) \leq (c+1) \times \text{size}(D_j) \times [1 + |\mathcal{C}| \times (2L^2 + L^4) \times 2d]$, where $d = \max(\{\tau(\mathcal{C}), \tau(\mathcal{D})\}) + 1$ is a bound on the term depth in $\mathcal{C}$ and $\mathcal{D}$, c denotes the maximum arity of the symbols in Σ, and L is an upper bound on the number of literals in the clauses $C_i \in \mathcal{C}$ and in the clauses $D_j \in \mathcal{D}$.
2. For all $i \in \{1, \ldots, n\}$, check by an oracle for first-order subsumption that $C_i \not\leq_s D^*$ holds.

The correctness of this algorithm follows immediately from Lemma 6.9. The polynomial time complexity is guaranteed by the polynomial bound on the size of the counterexample guessed in the first step. The first-order subsumption oracle is in NP (cf. [8, Problem LO18]). Hence, the overall algorithm is in Σ_2^p. ■

Together with the Π_2^p-hardness result from [23, Theorem 4.2], we may therefore conclude that the CLAUSE-H-SUBSUMPTION problem is Π_2^p-complete.

COROLLARY 6.1. *Let H be an arbitrary Herbrand universe with at least two elements. Then the CLAUSE-H-SUBSUMPTION problem over H is Π_2^p-complete.*

7. CONCLUSION AND FUTURE WORK

The focus of this work is on the complexity of several decision problems related to atomic representations of Herbrand models. We have proven the coNP-completeness of the model equivalence problem and of the clause evaluation problem for ARMs. Likewise, the coNP-completeness of the total cover problem and the atomic H-subsumption problem have been established in this paper. As a byproduct of the coNP-membership proof, we have provided an appropriate representation of the complement of an ARM. Finally, we have also shown the Π_2^p-membership of clausal H-subsumption over an arbitrary Herbrand universe H.

The complexity results may in a sense seem to be a bit discouraging. Nevertheless, the success of ARMs in the field of automated model building suggests that the search for reasonably efficient algorithms for these decision problems should not be given up. In [22], algorithms for atomic H-subsumption, model equivalence, and clause evaluation in ARMs are presented, which are much more efficient than the original ones in [7]. But there is clearly ample space for further improvement.

In [10], the following observation has led to a different approach to dealing with the high computational cost of these decision problems: For every ARM there is a large number of equivalent representations. However, as far as the actual cost of the algorithms in [7] and [22] is concerned, such equivalent representations may behave quite differently. Hence, it makes sense to identify desirable properties of atom sets and to transform an arbitrary ARM $\mathcal{A}$ into an equivalent ARM $\mathcal{A}'$ of the desired form. The cost of this kind of preprocessing step may be far outweighed by the time saved when the clause evaluation (or any other) algorithm is run repeatedly on the transformed atom set $\mathcal{A}'$ rather than on $\mathcal{A}$.

ACKNOWLEDGMENT

We thank the anonymous referees for suggesting some significant improvements.

REFERENCES

1. Baader, F., and Siekmann, J. H. (1994), Unification theory, *in* "Handbook of Logic in Artificial Intelligence and Logic Programming" (D. M. Gabbay, C. J. Hogger, and J. A. Robinson, Eds.), pp. 41–125, Oxford University Press, Oxford, UK.
2. Chang, C. L., and Lee, R. C. T. (1973), "Symbolic Logic and Mechanical Theorem Proving," Academic Press, New York.
3. Caferra, R., and Peltier, N. (1995), Extending semantic resolution via automated model building: Applications, *in* "Proceedings of IJCAI'95," pp. 328–334, Morgan Kaufmann, New York.
4. Caferra, R., and Zabel, N. (1991), Extending resolution for model construction, *in* "Proceedings of Logics in AI—JELIA'90," Lecture Notes in Artificial Intelligence, Vol. 478, pp. 153–169, Springer, New York.
5. Comon, H., and Lescanne, P. (1989), Equational problems and disunification, *J. Symbolic Comput.* **7**, 371–425.
6. Falaschi, M., Levi, G., Martelli, M., and Palamidessi, C. (1988), A new declarative semantics for logic languages, *in* "Proceedings of Joint International Conference on Logic Programming," pp. 993–1005.
7. Fermüller, C., and Leitsch, A. (1996), Hyperresolution and automated model building, *J. Logic and Comput.* **6**(2), 173–230.
8. Garey, M. R., and Johnson, D. S. (1979), "Computers and Intractability, A Guide to the Theory of NP-Completeness," Freeman, San Francisco.
9. Gottlob, G., Marcus, S., Nerode, A., Salzer, G., and Subrahmanian, V. S. (1996), A nonground realization of the stable and well-founded semantics, *Theoret. Comput. Sci.* **166**, 221–262.
10. Gottlob, G., and Pichler, R. (1998), Towards a compact knowledge representation by AR models, *in* "Proceedings of WLP'98 (13th Workshop on Logic Programming)"; also, Tech. Rep. 1843-1998-10, Technical University of Vienna, [http://www.kr.tuwien.ac.at/research/wlp98].
11. Gottlob, G., and Pichler, R. (1999), Working with ARMs: Complexity results on atomic representations of Herbrand models, *in* "Proceedings of LICS'99," pp. 306–315, IEEE Computer Society Press, Los Alamitos, CA.
12. Kapur, D., Narendran, P., Rosenkrantz, D., and Zhang, H. (1991), Sufficient-completeness, ground-reducibility, and their complexity, *Acta Informatica* **28**, 311–350.
13. Kunen, K. (1987), Answer sets and negation as failure, *in* "Proceedings of the Fourth International Conference on Logic Programming, Melbourne," pp. 219–228.
14. Kuper, G., McAloon, K., Palem, K., and Perry, K. (1988), Efficient parallel algorithms for antiunification and relative complement, *in* "Proceedings of LICS'88," pp. 112–120, IEEE Computer Society Press, Los Alamitos, CA.
15. Lassez, J.-L., Maher, M., and Marriott, K. (1991), Elimination of negation in term algebras, *in* "Proceedings of MFCS'91," Lecture Notes in Computer Science, Vol. 520, pp. 1–16, Springer, New York.
16. Lassez, J.-L., and Marriott, K. (1987), Explicit representation of terms defined by counter examples, *J. Automated Reasoning* **3**, 301–317.

17. Leitsch, A. (1997), "The Resolution Calculus," Texts in Theoretical Computer Science, Springer, New York.
18. Maher, M. (1988), Complete axiomatizations of the algebras of finite, rational and infinite trees, *in* "Proceedings of LICS'88," pp. 348–357, IEEE Computer Society Press, Los Alamitos, CA.
19. Maĺcev, A. (1961), On the elementary theories of locally free universal algebras, *Soviet Math. Dok.* **2**(3), 768–771.
20. Martelli, A., and Montanari, U. (1976), Unification in linear time and space: A structured presentation, Tech. Rep. B 76-16, University of Pisa.
21. Paterson, M., and Wegman, M. (1978), Linear unification, *J. Computer and System Sci.* **16**, 158–167.
22. Pichler, R. (1998), Algorithms on atomic representations of Herbrand models, *in* "Proceedings of Logics in AI - JELIA'98," Lecture Notes in Artificial Intelligence, Vol. 1489, pp. 199–215, Springer, New York.
23. Pichler, R. (1999), On the complexity of H-subsumption, *in* "Proceedings of CSL'98," Lecture Notes in Computer Science, Vol. 1584, pp. 355–371, Springer, New York.
24. Robinson, J. A. (1965), A machine oriented logic based on the resolution principle, *J. Assoc. Comput. Mach.* **12**(1), 23–41.
25. Tammet, T. (1991), Using resolution for deciding solvable classes and building finite models, *in* "Baltic Computer Science," Lecture Notes in Computer Science, Vol. 502, pp. 33–64, Springer, New York.
26. Vorobyov, S. (1996), An improved lower bound for the elementary theories of trees, *in* "Proceedings of CADE-13," Lecture Notes in Artificial Intelligence, Vol. 1104, pp. 275–287, Springer, New York.
27. Vorobyov, S., and Voronkov, A. (1998), Complexity of nonrecursive logic programs with complex values, *in* "Proceedings of PODS'98," pp. 244–253, ACM Press, New York.