Algorithmic Aspects of Model Representations

Reinhard Pichler

Technische Universität Wien, A-1040 Vienna, Austria
pichler@dbai.tuwien.ac.at

Abstract. Herbrand models play an important role in many areas of Computer Science like Logic Programming, Functional Programming, Machine Learning, etc. Moreover, models (in particular, Herbrand models) are at the very heart of Automated Model Building, which has evolved as an important subdiscipline of Automated Deduction over the past decade.
A crucial issue for dealing with models on the computer is the selection of an appropriate formalism for representing models. A desirable property of such a formalism is that (efficient) algorithms should exist for evaluating clauses in a model thus represented. Moreover, also algorithms for deciding the so-called model equivalence problem have received great interest.
In this survey, we recall three formalisms for representing Herbrand models, namely atomic representations of Herbrand models ([11]), constrained atoms ([6, 7]), and contexts ([2, 3]). Specifically, we shall recall recent results on algorithms and on the complexity of the aforementioned decision problems for these model representations.

1 Introduction

Herbrand models play an important role in many areas of Computer Science like Logic Programming, Functional Programming, Machine Learning, etc. Within the field of Automated Deduction, Automated Model Building has evolved as an important subdiscipline over the past decade (see [4]). Vivid interest in this topic is documented, e.g., by workshops at CADE-17 and CADE-19.[1] Clearly, models (in particular, Herbrand models) are at the very heart of the research in Automated Model Building. The applicability of models to the area of Automated Deduction is basically twofold:

- First, rather than just proving that some input formula is *not* a theorem, it would be desirable for a theorem prover to provide some insight as to *why* a given formula is not a theorem. To this end, the theorem prover tries to construct a counter-model rather than just giving the answer "NO".
- The second application of models arises from the idea of guiding the proof search by providing some additional knowledge on the domain from which the input formula is taken. This knowledge can be represented in the form of a model, which may then be used e.g. in semantic resolution.

In this survey, we concentrate on *Herbrand* models only. But of course, there are also other kinds of models that have received interest (e.g., finite models). In order to actually deal with models on a computer, we need an appropriate formalism for representing models. In case of finite models, such a representation may simply consist of a definition of the domain plus tables ('diagrams') for each predicate and function symbol, that record the value for each tuple of domain elements. On the other hand, Herbrand models are in general infinite objects. Hence, the question of how to represent them does not have such an obvious answer. Moreover, it should be noted that there are uncountably many Herbrand models over a given signature. Thus, only a subset of all such models can be represented by any given syntactic formalism.

Therefore, an important property of model representation formalisms is their *expressive power*, i.e. which models can be represented in one formalism as compared to another formalism. From

[1] See `http://www.uni-koblenz.de/~peter/CADE17-WS-MODELS/` and `http://www.uni-koblenz.de/~peter/models03/`, respectively, for the proceedings.

an algorithmic point of view, the following requirements on model representation formalisms are postulated in [11]: a model representation formalism should admit (efficient) algorithms for the following two decision problems:

- The CLAUSE EVALUATION problem: Given a representation M of a model and a clause C, does C evaluate to "true" in this model?
- The MODEL EQUIVALENCE problem: Given two representations M and N of models, do M and N represent the same model?

The importance of the first decision problem is obvious. Suppose that we want to use a model as an input to an automated theorem prover based on semantic resolution. Then an efficient CLAUSE EVALUATION algorithm is clearly indispensable. The significance of the second problem comes from the fact that formalisms presented in the model building literature for representing models usually allow for many different ways of representing the same model. However, when we actually want to compute the truth value of arbitrary clauses in such a model, then the efficiency of this computation depends to a large extent on the specific representation rather than just the model thus represented. It is therefore important to look for transformations of the model representation constructed in the first step into an equivalent one with "better" computational properties. But then we have to make sure, of course, that the model representation resulting from such a transformation is equivalent to the original one.

In this survey, we recall three formalisms for representing models, namely atomic representations of Herbrand models (in Section 3), constrained atoms (in Section 4), and contexts (in Section 5). Specifically, we shall recall recent results on algorithms and complexity of the aforementioned decision problems for these model representations. Moreover, in Section 2, we recall some basic notions and in Section 6, we give a conclusion.

2 Preliminaries

The reader is assumed to be familiar with the basic concepts of computational logic. In this section, we only recall the most important concepts for our further discussion. Let Σ denote a finite set of predicate symbols and function symbols, each with some arity $k \geq 0$. The set H of all ground terms over Σ is called the *Herbrand universe* over Σ. In the sequel, we usually consider the signature Σ as arbitrary but fixed when talking about some Herbrand universe H. We call H *non-trivial*, if it contains at least 2 elements.

A *literal* is either an atom A or a negated atom $\neg A$. *Clauses* are written as $C = L_1 \vee \cdots \vee L_n$, where the L_i's are literals. Recall that clauses are basically a short-hand notation for closed first-order formulae where all variables are universally quantified. A *Herbrand interpretation* of a clause (or a set of clauses) is an interpretation which interprets all ground terms "by themselves", so to speak. Hence, a Herbrand interpretation is given by the interpretation of the predicate symbols only. In particular, such an interpretation is uniquely determined by a subset of the Herbrand base, namely by those ground atoms over the given signature Σ which evaluate to "true".

In general, a *model* is an *interpretation* which validates a certain formula (or, analogously, a clause set). As long as one is concerned with the actual model construction, it is clear which formula is validated by the interpretation thus constructed. However, when one starts to work with such an interpretation in a different context (e.g. as input to a theorem prover based on semantic resolution), the connection between the interpretation and the formula which is validated by this interpretation is no longer obvious. In fact, it is a bit inaccurate to talk about "models" rather than "interpretations", when it is not clear, which formula is actually validated by a given interpretation. However, this kind of inaccuracy is very common in the model building literature. We shall, therefore, also refer to "interpretations" as "models" without having a particular formula in mind which is validated by such an interpretation.

3 Atomic Representations of Herbrand Models

In [11], Atomic Representations of Herbrand Models (= ARMs, for short) were introduced as finite sets $\mathcal{A} = \{A_1, \ldots, A_n\}$ of atoms, s.t. a ground atom evaluates to "true" in the Herbrand model represented by $\mathcal{A}$, iff it is a ground instance of some atom $A_i \in \mathcal{A}$.

Example 1. Let $\mathcal{A} = \{P(f(x), a), P(a, a), Q(x, x), Q(a, f(x))\}$ be an ARM and let the signature $\Sigma = \{P, Q, a, b, f\}$. Then the clause $C_1 = P(x, a) \vee \neg Q(x, f(a))$ evaluates to "true" in the model defined by $\mathcal{A}$. In order to see this, we have to verify, that all H-ground instances of C_1 evaluate to "true": To this end, we distinguish three kinds of possible values that the variable x can take and show that in each case at least one literal in C_1 evaluates to "true":

- *For $x = a$ the literal $P(a, a)$ evaluates to "true".*
- *Now let $x = b$. Then $Q(b, f(a))$ is not an instance of any atom in $\mathcal{A}$. Hence, $Q(b, f(a))$ evaluates to "false" and, thus, $\neg Q(b, f(a))$ evaluates to "true".*
- *Finally, let x be a term with leading symbol f, i.e., $x = f(t)$ for some term $t \in H$. Then $P(f(t), a)$ is an instance of $P(f(x), a) \in \mathcal{A}$ and, therefore, $P(f(t), a)$ evaluates to "true".*

On the other hand, the clause $C_2 = P(x, y) \vee \neg Q(a, y)$ evaluates to "false". This can be seen by showing that there exists at least one H-ground instance of C_2 that evaluates to "false". In fact, $P(a, f(a)) \vee \neg Q(a, f(a))$ is such a ground instance. It is easy to verify that both literals evaluate to "false". □

The key to the decision problems is yet another problem, which was called the ATOMIC H-SUBSUMPTION problem in [11], i.e.: Given an atom set $\mathcal{A} = \{A_1, \ldots, A_n\}$ and an atom B over some Herbrand universe H, is every H-ground instance of B an instance of some atom $A_i \in \mathcal{A}$? If this is the case, then we write $\mathcal{A} \leq_{sH} B$. In [11], the MODEL EQUIVALENCE problem and the CLAUSE EVALUATION problem are reduced to the (ATOMIC) H-SUBSUMPTION problem as follows:

Lemma 1. *Let $\mathcal{A} = \{A_1, \ldots, A_n\}$ and $\mathcal{B} = \{B_1, \ldots, B_m\}$ be ARMs w.r.t. some Herbrand universe H. Then $\mathcal{A}$ and $\mathcal{B}$ are equivalent, iff $\{A_1, \ldots, A_n\} \leq_{sH} B_j$ for every $j \in \{1, \ldots, m\}$ and $\{B_1, \ldots, B_m\} \leq_{sH} A_i$ for every $i \in \{1, \ldots, n\}$.*

In order to reduce also the CLAUSE EVALUATION problem to the H-SUBSUMPTION problem, we first have to recall that H-subsumption is not necessarily restricted to atoms, i.e.: For a clause set $\mathcal{C}$ and a clause D, we say that $\mathcal{C}$ *H-subsumes* D (written as $\mathcal{C} \leq_{sH} D$), iff every H-ground instance of D is subsumed by some clause $C \in \mathcal{C}$. More generally, if $\mathcal{D}$ is a clause set, we say that $\mathcal{C}$ *H-subsumes* $\mathcal{D}$ (written as $\mathcal{C} \leq_{sH} \mathcal{D}$), iff $\mathcal{C} \leq_{sH} D$ holds for every clause $D \in \mathcal{D}$. Then we have:

Lemma 2. *Let $\mathcal{A} = \{A_1, \ldots, A_n\}$ be an ARM w.r.t. some Herbrand universe H and let $C = L_1 \vee \cdots \vee L_l \vee \neg M_1 \vee \cdots \vee \neg M_m$ be a clause over H. Then we distinguish two cases:*

- Case 1*: $m = 0$, i.e.: C is a positive clause. Then C evaluates to "true" $\Leftrightarrow \mathcal{A} \leq_{sH} C$.*
- Case 2*: $m > 0$, i.e.: C contains at least one negative literal. Now let $\rho_h(\mathcal{A} \cup \{C\})$ denote the set of all hyperresolvents that are derivable from $\mathcal{A} \cup \{C\}$. Then C evaluates to "true" $\Leftrightarrow$ $\mathcal{A} \leq_{sH} \rho_h(\mathcal{A} \cup \{C\})$.*

Equivalent problems to the H-SUBSUMPTION problem have been studied in many areas of Computer science, such as in Machine Learning (cf. [19]) in *Logic Programming* (cf. [12], [18]), in Functional Programming (cf. [18]), etc. Consequently, many different approaches for deciding the H-SUBSUMPTION problem (or equivalent problems) have been presented in the literature. We only recall the algorithm from [11] here, which is based on the following property: *If $\mathcal{C}$ and $\mathcal{D}$ are sets of clauses, s.t. the minimum depth of variable occurrences in $\mathcal{D}$ is greater than the depth of $\mathcal{C}$, then H-subsumption and ordinary subsumption coincide.* Hence, the H-SUBSUMPTION problem $\mathcal{A} \leq_{sH} C$ for an atom set $\mathcal{A}$ and a clause C can be decided as follows: First, C is transformed into an equivalent clause set $\mathcal{C}'$ by partial saturation, s.t. the minimum depth of variable occurrences

in $\mathcal{C}'$ is greater than the depth of $\mathcal{A}$. Then an ordinary subsumption test $\mathcal{A} \leq_s C'$ is applied to every clause $C' \in \mathcal{C}'$.

As to the complexity of the decision problems studied here, it is convenient to consider also the TOTAL COVER problem, which is defined as follows: Given an atom set $\mathcal{A} = \{P(\boldsymbol{t_1}), \ldots, P(\boldsymbol{t_n})\}$ over some Herbrand universe H, is every H-ground atom $P(\boldsymbol{s})$ an instance of some $P(\boldsymbol{t_i}) \in \mathcal{A}$?

It is easy to show the following relations between these problems (cf. [13, 14]): The TOTAL COVER problem can be reduced to the MODEL EQUIVALENCE problem, which in turn can be reduced to the ATOMIC H-SUBSUMPTION. Finally, ATOMIC H-SUBSUMPTION can be reduced to the CLAUSE EVALUATION problem. All these reductions are possible in polynomial time. In other words, TOTAL COVER is the "easiest" and CLAUSE EVALUATION is the "hardest" of these four problem. The inherent complexity of the TOTAL COVER problem (or equivalent problems) has been investigated by several authors independently, who proved its coNP-completeness (cf. [15], [16], [17]). Together with the coNP-membership of clause evaluation (cf. [13, 14]), we get the following result:

Theorem 1. *The following decision problems are coNP-complete over any non-trivial Herbrand universe:* TOTAL COVER*,* MODEL EQUIVALENCE *of ARMs,* ATOMIC H-SUBSUMPTION*, and* CLAUSE EVALUATION *of ARMs.*

4 Constrained Atoms

ARMs have a somehow restricted expressive power. In particular, they are not closed under complement, i.e.: Let $\mathcal{A} = \{A_1, \ldots, A_n\}$ be a set of atoms. Then, in general, the set of ground atoms that are *not* instances of the atoms $A_i \in \mathcal{A}$ cannot be expressed in terms of an ARM itself. In [6, 7], the expressive power of ordinary clauses is increased by adding *equational constraints*. A constrained clause (= c-clause, for short) over some Herbrand universe H is a construct of the form $[\![c : \mathcal{P}]\!]$, where c is a clause and $\mathcal{P}$ is an equational formula over H. An H-ground clause $c\sigma$ is an instance of $[\![c : \mathcal{P}]\!]$, iff σ is a solution of $\mathcal{P}$. Standard clauses can be considered as a special case of constrained clauses with the trivially true constraint $\top$.

Example 2. Let $\mathcal{A} = \{P(x, x)\}$ be an ARM. It can be shown (see [19]), that the complement of $\mathcal{A}$ does not have a representation by an ARM. In contrast, such a representation is easy by using equational constraints, namely: $\mathcal{A}' = \{[\![P(u, v) : u \neq v]\!]\}$. Note that $P(x, x)$ can also be considered as a constrained atom, namely $[\![P(x, x) : \top]\!]$. □

Actually, in [6, 7], models are defined in a slightly different way. Rather than just specifying the set of ground atoms that evaluate to "true" (and requiring that all other ground atoms evaluate to "false"), Caferra et al. introduced so-called *partial interpretations definable by equational formulae* (= peq-interpretations, for short). Such a peq-interpretation is given through a finite set $\mathcal{L} = \{[\![l_1 : \mathcal{P}_1]\!], \ldots, [\![l_n, \mathcal{P}_n]\!]\}$ of constrained literals (the l_i's are either atoms or negated atoms). A ground atom A evaluates to "true" in the interpretation defined by $\mathcal{L}$, iff A is an instance of some (positive) c-literal in $\mathcal{L}$. On the other hand, A evaluates to "false", if $\neg A$ is an instance of some (negative) c-literal in $\mathcal{L}$. Otherwise, the truth value of A is undefined. Of course, one has to make sure, that there exists no atom A, s.t. both A and $\neg A$ are instances of elements in $\mathcal{L}$.

The definition of the truth value of a negated ground atom $\neg A$ is obvious, i.e.: $\neg A$ is "true", iff A is "false". Likewise, $\neg A$ is "false", iff A is "true". Finally, $\neg A$ is "undefined", iff A is "undefined". The truth value of arbitrary c-clauses in a peq-interpretation is defined as follows: Let $C = M_1 \vee \cdots \vee M_k$ denote a ground clause. Then the truth value $I(C)$ in the peq-interpretation I represented by $\mathcal{L}$ is defined as follows:

$$I(C) = \begin{cases} \text{"true"} & \text{if } \exists i\text{: } I(M_i) = \text{"true"} \\ \text{"false"} & \text{if } \forall i\text{: } I(M_i) = \text{"false"} \\ \text{"undefined"} & \text{otherwise} \end{cases}$$

Now let $[\![c : \mathcal{Q}]\!]$ be an arbitrary c-clause. Then the truth value $I([\![c : \mathcal{Q}]\!])$ is defined as follows:

$$I([\![c : \mathcal{Q}]\!]) = \begin{cases} \text{“true”} & \text{if } \forall\ H\text{-ground instances } c\sigma \text{ of } [\![c : \mathcal{Q}]\!]\text{: } I(c\sigma) = \text{“true”} \\ \text{“false”} & \text{if } \exists\ H\text{-ground instance } c\sigma \text{ of } [\![c : \mathcal{Q}]\!]\text{: } I(c\sigma) = \text{“false”} \\ \text{“undefined”} & \text{otherwise} \end{cases}$$

Example 3. Let the peq-interpretation I be given through the set $\mathcal{L} = \{[\![P(x, f(y)) : x \neq y]\!], [\![\neg P(x, y) : x \neq a \wedge (\forall z) y \neq f(z)]\!]\}$ of c-literals and let the signature $\Sigma = \{P, a, b, f\}$. It is easy to check that $\{[\![P(x, f(y)) : x \neq y]\!]$ and $[\![\neg P(x, y) : x \neq a \wedge (\forall z) y \neq f(z)]\!]$ have no H-ground instances in common. Hence, the peq-interpretation I is well-defined.

Now consider the clause $C = P(x, x) \vee \neg P(a, f(x))$ or, equivalently, the c-clause $C' = [\![P(x, x) \vee \neg P(a, f(x)) : \top]\!]$. Then these clauses evaluate to “false” in I, since the H-ground instance $P(b, b) \vee \neg P(a, f(b))$ does. Indeed, $P(b, b)$ evaluates to “false”, since $\neg P(b, b)$ is an instance $[\![\neg P(x, y) : x \neq a \wedge (\forall z) y \neq f(z)]\!]$. Likewise, the second literal $\neg P(a, f(b))$ evaluates to “false”, since $P(a, f(b))$ is an instance of $[\![P(x, f(y)) : x \neq y]\!]$. □

The above condition for an arbitrary c-clause to evaluate to “true” can be expressed as the validity of an equational formula in the following way (cf. [5]):

Definition 1. *Let the peq-interpretation I over H be given through the set $\mathcal{L} = \{[\![L_1(\boldsymbol{s}_1) : \mathcal{P}_1]\!], \ldots, [\![L_n(\boldsymbol{s}_n) : \mathcal{P}_n]\!]\}$ of c-literals and let $C = [\![M_1(\boldsymbol{t}_1) \vee \cdots \vee M_k(\boldsymbol{t}_k) : \mathcal{Q}]\!]$ be a c-clause over H, where the L_i's and M_j's denote literal symbols (i.e., either unnegated or negated predicate symbols). Moreover, let $\boldsymbol{y}_i = Var([\![L_i(\boldsymbol{s}_i) : \mathcal{P}_i]\!])$ and suppose that $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_n$, $Var(C)$ are pairwise disjoint. Then the equational formula $\mathcal{F}_I(C)$ is defined as follows:*

$$\mathcal{F}_I(C) \equiv \mathcal{Q} \wedge \bigvee_{M_j = L_i} (\exists \boldsymbol{y}_i)[\mathcal{P}_i \wedge \boldsymbol{s}_i = \boldsymbol{t}_j]$$

Then the following condition holds:

Lemma 3. *Let I, $\mathcal{L}$, C, and $\mathcal{F}_I(C)$ be defined as above. Moreover, let $C\sigma$ be an arbitrary H-ground instance of C. Then the following equivalence holds:*

$$C\sigma \text{ evaluates to “true” in } I \Leftrightarrow \sigma \text{ is a solution of } \mathcal{F}_I(C)$$

But then the condition that C evaluates to “true” in I is clearly equivalent to the condition that $\mathcal{F}_I(C)$ and $\mathcal{Q}$ are equivalent. This is the case, iff the equational formula $\mathcal{F}_I(C) \leftrightarrow \mathcal{Q}$ is valid.

Likewise, the MODEL EQUIVALENCE problem can of course be reduced to the validity problem of equational formulae. This can be easily seen by first reducing the MODEL EQUIVALENCE problem to the CLAUSE EVALUATION problem and then applying the problem reduction via the formula $\mathcal{F}_I(C)$ from Definition 1:

Lemma 4. *Let the peq-interpretations I and J over H be given through the sets of c-literals $\mathcal{L} = \{[\![L_1(\boldsymbol{s}_1) : \mathcal{P}_1]\!], \ldots, [\![L_l(\boldsymbol{s}_l) : \mathcal{P}_l]\!]\}$ and $\mathcal{M} = \{[\![M_1(\boldsymbol{t}_1) : \mathcal{Q}_1]\!], \ldots, [\![M_n(\boldsymbol{t})_m) : \mathcal{Q}_m]\!]\}$, respectively.*

Then the peq-interpretations I and J are equivalent, iff every c-literal $[\![L_i(\boldsymbol{s}_i) : \mathcal{P}_i]\!] \in \mathcal{L}$ is “true” in J and every c-literal $[\![M_j(\boldsymbol{t}_j) : \mathcal{Q}_j]\!] \in \mathcal{M}$ is “true” in I.

Equational formulae have been studied by various authors (e.g., see [8], [9], [16], [20], [21], [24], [25], [26]). Several decision methods for the validity problem of equational formulae have been proposed. Unfortunately, they all have a very high computational complexity. But this cannot be helped by the following result from [25]:

Theorem 2. *The validity problem of equational formulae over an infinite Herbrand universe (i.e., where the signature contains at least one proper function symbol) is non-elementary recursive.*

Of course, we cannot expect to do better for the CLAUSE EVALUATION problem and the MODEL EQUIVALENCE problem of peq-interpretations. We thus have

Corollary 1. *The* CLAUSE EVALUATION *problem and the* MODEL EQUIVALENCE *problem of peq-interpretations over an infinite Herbrand universe is non-elementary recursive.*

Actually, in case of a finite Herbrand universe, the validity problem of equational formulae is "only" PSPACE-complete (cf. [16]). But of course, in the area of automated model building, the infinite case is far more relevant.

5 Contexts

Recently Baumgartner and Tinelli [1–3] have introduced a calculus for clause logic, called *model evolution*, that relies heavily on a particular form of model representation, namely the so-called "contexts". In [1–3], contexts are considered over a signature containing **infinitely many constant symbols**. Hence, throughout this section, we also assume that Σ contains infinitely many constant symbols.

A context is simply a finite set of literals. However, an important ingredient is the distinction between 'universal variables' and 'parameters'. The latter can be viewed as a form of variables, that do not necessarily admit instantiation by arbitrary ground terms; whereas universal variables can be understood as placeholders for arbitrary terms of the Herbrand universe (This idea will become clear in Definition 3 below). Each non-ground literal in a context either contains only universal variables or only parameters. Correspondingly, we speak of *universal literals* and *parameter literals*, respectively.

In addition to ordinary literals, every context also contains the *pseudo-literal* $\neg v$. Every negative literal is considered a proper instance of $\neg v$. The *pseudo-literal* $\neg v$ is used to guarantee that all atoms that are not explicitly specified to be true are false by default in contexts.

Analogously to peq-interpretations, contexts have to be non-contradictory in the sense that a ground atom cannot at the same time be true and false in the model thus represented. Let $\overline{L}$ denote the literal that is dual to L; i.e., $\overline{A} = \neg A$ and $\overline{\neg A} = A$. Then we have the following definitions:

Definition 2. *A* context Λ *is a finite set of literals containing the pseudo-literal* $\neg v$. Λ *is* contradictory *iff* $L\sigma = \overline{K}\sigma$ *for some variants* L, K *of elements in* Λ *where the substitution* σ *restricted to the parameters is a renaming.*

Definition 3. *Let* Λ *be a non-contradictory context. A ground atom* A *is true in the model* $\mathcal{M}(\Lambda)$ *represented by* Λ *iff one the following conditions holds:*

1. *A is an instance of some universal literal in Λ, or*
2. *A is an instance of a parameter literal $L \in \Lambda$, but $\neg A$ is not an instance of a literal $\neg B \in \Lambda$, where B is either universal or a proper instance of L.*

In other words, by the first condition, all instances of universal atoms in Λ are true. On the other hand, a ground instance A of a parameter atom L in Λ is true only if $\neg A$ is not an instance of a more specific literal or of a universal literal in Λ.

In accordance with [2], we write x, y, z to denote variables, and u, v, w for parameters. The following examples will help to illustrate the above definitions:

Example 4. The context $\Lambda_1 = \{\neg v, P(x, f(y)), \neg P(a, z)\}$ is contradictory, since $P(x, f(y))$ and $P(a, z)$ are unifiable, and the corresponding unifier does not instantiate any parameters (since no parameters at all occur in the atoms). However the context $\Lambda_2 = \{\neg v,\ P(x, f(y)), \neg P(a, u)\}$ is non-contradictory since, for all substitutions σ such that $P(x, f(y))\sigma = P(a, u)\sigma$ holds, the parameter u has to be instantiated. Likewise the (pseudo-)parameter v has to be instantiated when unified with $P(x, f(y))$. □

Example 5. Consider the context $\Lambda = \{\neg v,\ P(c,x,y),\ P(u,a,v),\ \neg P(w,w,b),\ P(u,v,b)\}$. By definition, all ground instances of (the universal atom) $P(c,x,y)$ are true in the model $\mathcal{M}(\Lambda)$. Likewise, the ground instance $P(a,a,a)$ of $P(u,a,v)$ is true $\mathcal{M}(\Lambda)$. However, since $\neg P(w,w,b)$ is a proper instance of $\neg P(u,v,b)$, the ground instance $P(d,d,b)$ of $P(u,v,b)$ is not true in $\mathcal{M}(\Lambda)$. For the ground atoms $P(a,a,b)$ and $P(c,c,b)$ the situation might look similar, i.e., they are instances of $P(u,v,b)$ and also of the more specific atom $P(w,w,b)$. Nevertheless, both $P(a,a,b)$ and $P(c,c,b)$ are true in $\mathcal{M}(\Lambda)$ – due to the atoms $P(u,a,v) \in \Lambda$ and $P(c,x,y) \in \Lambda$, respectively. Actually, it is easy to check that the model representetd by Λ contains exactly the ground atoms $\mathcal{M}(\Lambda) = \{P(r,s,t) \mid r = c \vee s = a \vee (r \neq s \wedge t = b)\}$. □

It turns out that the above recalled contexts are closely related to the more classical idea of "disjunctions of implicit generalizations" (DIGs, for short). Following ideas in [19], DIGs can be defined as follows:

Definition 4. *An* implicit generalization Γ *is an expression of the form $A/\mathcal{B}$, where A is an atom and $\mathcal{B}$ is a finite set of atoms. We simply write A for $A/\{\}$. Every ground atom that is an instance of A, but not an instance of any $B \in \mathcal{B}$ is said to be* contained *in $A/\mathcal{B}$.*

A disjunction of implicit generalizations Δ *(shortly:* DIG*) is defined as an expression of the form $A_1/\mathcal{B}_1 \sqcup \ldots \sqcup A_n/\mathcal{B}_n$, also written as $\bigsqcup_{1 \leq i \leq n} A_i/\mathcal{B}_i$. A ground atom is said to be* contained *in Δ if it is contained in $A_i/\mathcal{B}_i$ for some $i \in \{1,\ldots,n\}$. The model $\mathcal{M}(\Delta)$ represented by a DIG Δ is the set of all ground atoms that are contained in Δ.*

In [10], the following relation between contexts and DIGs is shown:

Theorem 3. *Contexts and DIGs have the same expressive power; i.e., a model $\mathcal{N}$ can be represented by a context $\Leftrightarrow$ $\mathcal{N}$ can be represented by a DIG.*

Proof. (Sketch). We only recall the proof of the "$\Rightarrow$" direction: Let $\Lambda = \{\neg v\} \cup \Lambda_u^+ \cup \Lambda_p^+ \cup \Lambda_u^- \cup \Lambda_p^-$ be a context, where Λ_u^+ are the positive universal literals, Λ_p^+ are the positive parameter literals, Λ_u^- are the negative universal literals, and Λ_p^- are the negative parameter literals of Λ, respectively. W.l.o.g., we may assume that all literals in Λ are pairwise variable-disjoint. Then the model $\mathcal{M}(\Lambda)$ represented by Λ coincides with the model $\mathcal{M}(\Delta_\Lambda)$ represented by the following DIG Δ_Λ:

$$\Delta_\Lambda = \bigsqcup_{K \in \Lambda_u^+} K \ \sqcup \bigsqcup_{K \in \Lambda_p^+} K/\big(\{\overline{L} \mid L \in \Lambda_p^-, \overline{L} \text{ is a proper instance of } K\} \cup \{\overline{L} \mid L \in \Lambda_u^-\}\big)$$

The property $\mathcal{M}(\Delta_\Lambda) = \mathcal{M}(\Lambda)$ is an immediate consequence of Definition 3. □

The complexity of the transition from contexts to DIGs and vice versa is classified in [10] as follows:

Theorem 4. *For any context Λ, an equivalent DIG Δ with $\mathcal{M}(\Lambda) = \mathcal{M}(\Delta)$ can be computed in polynomial time.*

In contrast, there exists a sequence Δ_n ($n > 1$) of DIGs, where the size of Δ_n is polynomial (in n), such that all contexts representing the same model as Δ_n are of exponential size (in n).

Proof. (Sketch). The first part of the theorem follows immediately from the construction of Δ_Λ in the proof of Theorem 3 (which only involves some instance checks and thus clearly is polynomial).

The second part is established via the following sequence Δ_n:

Let $\Delta_n = \bigsqcup_{1 \leq i \leq n} P(u_1,\ldots,u_n)/\{P(u_1,\ldots,u_{i-1},0,u_{i+1},\ldots,u_n),$
$P(u_1,\ldots,u_{i-1},1,u_{i+1},\ldots,u_n)\},$

where the u_i ($1 \leq i \leq n$) are pairwise distinct parameters. It can be shown that every context equivalent to Δ_n must contain the set $\Lambda_n^- = \{\neg P(d_1,\ldots,d_n) \mid d_i \in \{0,1\}, 1 \leq i \leq n\}$ of negative literals. Moreover, $|\Lambda_n^-| = 2^n$ clearly holds. □

In other words, DIGs and contexts have the same expressive power for representing models – even though DIGs may be exponentially more succinct. Note that DIGs have long been known to be equivalent to *'constrained atoms'*. In fact, translating DIGs into sets of constrained atoms is straightforward; the translation of constrained atoms into DIGs can be done effectively via results in [20].[2] Note that the equivalence of contexts and DIGs with constrained atoms implies that the set of models representable by contexts and DIGs is also closed under complement.

For the decision problems studied here, the following comparatively favourable results are shown in [10]:

Theorem 5. *The decision problems* CLAUSE EVALUATION *and* MODEL EQUIVALENCE *are coNP-complete both for contexts and for DIGs.*

6 Conclusion

In this survey, we have recalled some algorithmic and complexity theoretical aspects of three model representation formalisms, namely: atomic representations of Herbrand models, constrained atoms, and contexts. However, many more formalisms for representing models can be found in the literature like (various forms of) term schematizations, (linear) atoms with positional constraints, term grammars and finite tree automata, tree automata with brotherhood constraints, etc. A good overview with a detailed comparison of the expressive power of these formalisms is given in [22, 23].

Efficient CLAUSE EVALUATION and MODEL EQUIVALENCE algorithms are an important issue, if one wants to work with models after they have been constructed. Hence, a thorough complexity analysis and the search for reasonably efficient algorithms also for other model representation formalisms is an interesting task for future work in this area.

As has already been mentioned, Herbrand models play an important role also in other areas of Computer Science, like Logic Programming, Functional Programming, Machine Learning. In some cases, it has turned out that similar representation formalisms and similar algorithmic properties have been investigated independently in various areas (cf. comments on atomic representations in Section 3). However, a unified picture of various representation formalisms and properties studied in different areas is missing.

References

1. P. Baumgartner, A. Fuchs, and C. Tinelli. Darwin: A theorem prover for the model evolution calculus. In *Proc. ESFOR'04*, Electronic Notes in Theoretical Computer Science. Elsevier, 2004.
2. P. Baumgartner and C. Tinelli. The model evolution calculus. In *Proc. CADE-19*, volume 2741 in LNCS, pages 350–364. Springer Verlag, 2003.
3. P. Baumgartner and C. Tinelli. The model evolution calculus. Fachberichte Informatik, 1/2003, Universität Koblenz Landau, 2003. Extended Version of [2].
4. R. Caferra, A. Leitsch, and N. Peltier. *Automated Model Building*, volume 31 of *Applied Logic Series*. Kluwer Academic Publishers, 2004.
5. R. Caferra and N. Peltier. Decision procedures using model building techniques. In *Proc. CSL'95*, volume 1092 of *LNCS*, pages 130–144. Springer Verlag, 1995.
6. R. Caferra and N. Zabel. Extending resolution for model construction. In *Proc. JELIA'90*, volume 478 of *LNCS*, pages 153–169. Springer Verlag, 1991.
7. R. Caferra and N. Zabel. A method for simultanous search for refutations and models by equational constraint solving. *Journal of Symbolic Computation*, 13:613–642, 1992.
8. H. Comon and C. Delor. Equational formulae with membership constraints. *Information and Computation*, 112(2):167–216, 1994.
9. H. Comon and P. Lescanne. Equational problems and disunification. *Journal of Symbolic Computation*, 7(3/4):371–425, 1989.

[2] Of course, the latter translation has non-elementary complexity in the worst-case, if arbitrary quantifier alternations are allowed in the constraining formula, see [25]).

10. C. Fermüller and R. Pichler. Model representation via contexts and implicit generalizations. In *Proc. CADE-20*, volume 3632 in LNCS, pages 409–423. Springer Verlag, 2005.
11. C. G. Fermüller and A. Leitsch. Hyperresolution and automated model building. *Journal of Logic and Computation*, 6(2):173–230, 1996.
12. G. Gottlob, S. Marcus, A. Nerode, G. Salzer, and V. S. Subrahmanian. A non-ground realization of the stable and well-founded semantics. *Theoretical Computer Science*, 166:221–262, 1996.
13. G. Gottlob and R. Pichler. Working with ARMs: Complexity results on atomic representations of Herbrand models. In *Proc. LICS'99*, pages 306–315. IEEE Computer Society, 1999.
14. G. Gottlob and R. Pichler. Working with ARMs: Complexity results on atomic representations of Herbrand models. *Information and Computation*, 165:183–207, 2001.
15. D. Kapur, P. Narendran, D. Rosenkrantz, and H. Zhang. Sufficient-completeness, ground-reducibility and their complexity. *Acta Informatica*, 28(4):311–350, 1991.
16. K. Kunen. Answer sets and negation as failure. In *Proc. ICLP'87*, pages 219–228. MIT Press, 1987.
17. G. Kuper, K. McAloon, K. Palem, and K. Perry. Efficient parallel algorithms for anti-unification and relative complement. In *Proc. LICS'88*, pages 112–120. IEEE Computer Society, 1988.
18. J.-L. Lassez, M. Maher, and K. Marriott. Elimination of negation in term algebras. In *Proc. MFCS'91*, volume 520 in LNCS, pages 1–16. Springer Verlag, 1991.
19. J.-L. Lassez and K. Marriott. Explicit representation of terms defined by counter examples. *Journal of Automated Reasoning*, 3(3):301–317, 1987.
20. M. Maher. Complete axiomatizations of the algebras of finite, rational and infinite trees. In *Proc. LICS'88*, pages 348–357. IEEE Computer Society, 1988.
21. A. Maĺcev. On the elementary theories of locally free universal algebras. *Soviet Mathematical Doklady*, 2(3):768–771, 1961.
22. R. Matzinger. Comparing computational representations of Herbrand models. In *Proc. KGC'97*, volume 1289 in LNCS, pages 203–218. Springer Verlag, 1997.
23. R. Matzinger. *Computational Representations of Models in First-Order Logic*. PhD thesis, Vienna University of Technology, 2000.
24. R. Pichler. Solving equational problems efficiently. In *Proc. CADE-16*, volume 1632 in LNAI, pages 97 – 111. Springer Verlag, 1999.
25. S. Vorobyov. An improved lower bound for the elementary theories of trees. In *Proc. CADE-13*, volume 1104 in LNAI, pages 275–287. Springer Verlag, 1996.
26. S. Vorobyov and A. Voronkov. Complexity of nonrecursive logic programs with complex values. In *Proc. PODS'98*, pages 244–253. ACM Press, 1998.