Proceedings, Fifteenth International Conference on
Principles of Knowledge Representation and Reasoning (KR 2016)

Implicit Hitting Set Algorithms for Reasoning Beyond NP

Paul Saikko and **Johannes P. Wallner** and **Matti Järvisalo**
HIIT, Department of Computer Science, University of Helsinki, Finland

Abstract

Lifting a recent proposal by Moreno-Centeno and Karp, we propose a general framework for so-called implicit hitting set algorithms for reasoning beyond NP. The framework is motivated by empirically successful specific instantiations of the approach—based on interactions between a Boolean satisfiability (SAT) solver and an integer programming (IP) solver—in the context of maximum satisfiability (MaxSAT). The framework opens up opportunities for developing implicit hitting set algorithms for various important reasoning problems in KR by implementing domain-specific reasoning modules with SAT and IP solvers. We detail instantiations of the framework for the minimum satisfiability problem—as a natural dual of MaxSAT—and, as a central KR problem, for propositional abduction, covering the second level of the polynomial hierarchy. We show empirically that an implementation of the instantiation for propositional abduction surpasses the efficiency of an approach based on encoding and solving propositional abduction instances as disjunctive logic programming under answer set semantics. We also study key properties of the general framework.

1 Introduction

A great majority of important decision and optimization problems in knowledge representation and reasoning (KR) and artificial intelligence are notoriously hard. In fact, variants of various central KR problems, such as propositional circumscription, abduction, belief revision, and others (Eiter and Gottlob 1992; 1993; 1995b; Stillman 1992; 1990; Gottlob 1992; Eiter and Lukasiewicz 2000) are hard at least for the second level of the polynomial hierarchy, and thus presumably go beyond NP. While NP-hard decision and optimization problems are in the classical sense intractable, the rise of surprisingly effective constraint solving technology, including e.g. Boolean satisfiability (SAT) and integer programming (IP) solvers, enables finding optimal solutions to complex NP-hard real-world problems in a variety of domains. Furthermore, the use of SAT solvers as practical NP oracles has proven a promising approach to solving decision problems beyond NP.

In this work, we propose a general framework for solving reasoning tasks beyond NP. The framework builds on the idea of so-called *implicit hitting set algorithms*, as proposed recently by (Moreno-Centeno and Karp 2013), with an emphasis on classical NP-complete decision problems. As outlined in (Moreno-Centeno and Karp 2013), implicit hitting set algorithms work by iteratively ruling out an increasing set of non(-optimal) solutions from further consideration by obtaining hitting sets over the non-solutions, until an actual (provably optimal) solution is found. From the practical perspective, the framework is motivated by empirically successful applications in the context of maximum satisfiability (Davies and Bacchus 2011; 2013b; 2013a), where the MaxSAT solvers MaxHS and LMHS, implementing implicit hitting set algorithms based on interacting SAT and IP solvers, have taken top positions in recent MaxSAT Evaluations (Argelich et al. 2015).

Motivated by this success, in this work we outline a general framework for implicit hitting set algorithms. Specifically (but by no means restricted to), the framework is developed with instantiations based on SAT and IP solvers in mind; the SAT solver acts (or, going beyond NP, multiple SAT solvers act) the role of a "core extractor" used for extracting non-solutions, and the IP solver acts as a hitting set optimizer, used for ruling out the thus far found non-solutions from further consideration. The framework thus provides novel algorithms for a variety of hard reasoning tasks via modularly instantiating the core extraction and hitting set modules in domain-specific ways via SAT and IP solvers specifically well-suited for the respective tasks of providing proofs of unsatisfiability and optimization. We detail novel instantiations of the general framework, using as a running example minimum satisfiability, the dual of MaxSAT that has recently received increasing attention, and, most interestingly, going beyond NP, for the problem of propositional abduction that is hard for the second-level of the polynomial hierarchy. To illustrate the practical potential of the framework, we present results from an empirical evaluation of a prototype implementation of the instantiation for abduction. We show empirically that the implementation for propositional abduction surpasses the efficiency of an approach based on encoding and solving propositional abduction instances as disjunctive logic programming under answer set semantics (Brewka, Eiter, and Truszczyński 2011). Furthermore, from a more theoretical perspective, we discuss fundamental properties and correctness of the general framework, as well as motivate the instantiation for abduction via parameterized complexity arguments.

The rest of this paper is organized as follows. After an overview of maximum satisfiability and the so-called MaxHS implicit hitting set approach to solving MaxSAT (Section 2), we describe the proposed general framework for implicit hitting set algorithms and detail key properties of the framework (Section 3). We then describe in detail how the framework can be instantiated for propositional abduction (Section 4), and present an empirical evaluation of the efficiency of a prototype implementation of the instantiation (Section 5). Finally, we discuss related work in detail (Section 6).

2 A Hitting Set Approach to MaxSAT

The basis for the implicit hitting set framework is the recent successful MaxHS algorithm for MaxSAT by Davies and Bacchus (2011; 2013b; 2013a). The novelty of this algorithm, compared to other successful MaxSAT algorithms, is its hybrid nature. The key idea behind MaxHS is to separate the satisfiability and optimization parts of MaxSAT so that a suitable method can be used to solve each part: an integer programming (IP) solver generates candidate sets of soft clauses by *hitting* a set of constraints in an optimal way, while a state-of-the-art SAT solver tries to find a satisfying assignment for the selected clauses.

Maximum Satisfiability

A literal is a variable y or a negated variable $\neg y$. A clause is a disjunction of literals. A propositional formula in conjunctive normal form (CNF) is a conjunction of clauses. A CNF formula can be viewed as a set of clauses. A truth assignment is a function τ from variables to $\{0,1\}$, denoting false and true, respectively. Satisfaction of a truth assignment for a formula is defined as usual.

An instance $\varphi = (\varphi_h, \varphi_s, c)$ of the (Weighted) *Partial MaxSAT* problem consists of a formula φ_h in CNF, to which we refer to also as *hard* clauses, a set φ_s of *soft* clauses, and a cost function $c : \varphi_s \rightarrow \mathbb{R}^+$. Any truth assignment τ that satisfies every clause in φ_h is a *solution* to φ. The cost of a solution τ to φ for the Partial MaxSAT problem is $\sum_{x\in\varphi_s} c(x)\cdot(1-\tau(x))$, i.e., the total cost of soft clauses not satisfied by τ. A solution τ is *optimal* for φ if $\textsc{cost}(\varphi,\tau) \leq \textsc{cost}(\varphi,\tau')$ holds for any solution τ' to φ. Given φ, the task for the Partial MaxSAT problem is to find an optimal solution to φ. From here on, we refer to Partial MaxSAT simply as MaxSAT.

Hitting Sets

For a set X, let $K \subseteq 2^X$ be a set of subsets of X. A hitting set $S \subseteq X$ of K is a set that intersects (hits) each $E \in K$, i.e., $S \cap E \neq \emptyset$ for all $E \in K$. For $K \subseteq 2^X$, the set of all hitting sets of K is $\mathrm{HS}(K) = \{S \subseteq X \mid S \text{ hitting set of } K\}$. Attaching costs to elements in X by a function $c : X \rightarrow \mathbb{R}^+$, the set of minimum-cost hitting sets is $\mathrm{HS}_c(K) = \operatorname*{arg\,min}_{S\in\mathrm{HS}(K)}(c(S))$, with $c(S) = \sum_{s\in S} c(s)$.

The MaxHS Algorithm

A basic concept employed by the MaxHS algorithm is that of an (unsatisfiable) core of a MaxSAT instance. An unsatisfiable subset, or unsatisfiable core, of a MaxSAT instance $\varphi = (\varphi_h, \varphi_s, c)$ is a set of clauses $C \subseteq \varphi_s$ such that $\varphi_h \wedge \bigwedge_{x\in C} x$ is unsatisfiable.

The MaxHS algorithm is based on the fact that a minimum-cost hitting set for the unsatisfiable cores of a MaxSAT instance corresponds to the set of falsified clauses of an optimal MaxSAT solution. The intuition behind this is simple: if we know every possible unsatisfiable subset of clauses and a minimum-cost hitting set over these subsets, there must exist a MaxSAT solution which satisfies every clause not in the hitting set. Furthermore, because the hitting set has minimum cost, this MaxSAT solution is optimal among all solutions.

Figure 1 (left) illustrates the MaxHS algorithm. At each iteration, a SAT solver is used to extract a new core of φ_h conjoined with clauses in $(\varphi_s \setminus S)$ until the hitting set S for the resulting set of cores corresponds to an optimal solution. When such an S is found, the SAT solver will return satisfiable, and the found satisfying variable assignment is an optimal solution to the MaxSAT instance. The minimum-cost hitting set problem can be solved with an IP solver. The IP solver only considers the set of cores found so far (which can be incrementally added to the problem as they are found) and the MaxSAT cost function; it does not require knowledge of the CNF formula. The SAT solver, on the other hand, operates independently of the cost function.

3 A General Framework for Implicit Hitting Set Procedures

In this section we generalize the MaxHS algorithm and its components into a general framework for developing implicit hitting set algorithms. As a core ingredient we need a general problem setting suitable for many contexts, which we introduce next. We assume a set of (domain) elements L and a predicate p over subsets $S \in 2^L$. The purpose of predicate p is to define solutions for the problem represented by p, i.e., if $p(S)$ holds for $S \subseteq L$, then S is a solution. Finally, a given cost function $c : L \rightarrow \mathbb{R}^+$ induces minimum-cost solutions. The function straightforwardly generalizes to sets $S \subseteq L$ by returning the sum, i.e., we have $c(S) = \sum_{s\in S} s$. We note that the general framework developed in this paper also allows for more sophisticated cost functions, e.g. non-linear functions, as long as $c(S) < c(S')$ whenever $S \subset S'$.

Definition 1. *Let L be a finite set, p a predicate over 2^L, and $c : L \rightarrow \mathbb{R}^+$. We call $P = (p, L, c)$ a minimization problem. The solutions of P are given by $Sol(P) = \{S \subseteq L \mid p(S) \text{ holds}\}$. Further, we define minimum-cost solutions as $Sol_c(P) = \operatorname*{arg\,min}_{S\in Sol(P)}(c(S))$.*

For the rest of the paper, we will focus on minimization problems; we note that the treatment naturally applies also to maximization problems, essentially by inverting the cost function in the standard way.

In detailing the general framework, we will exemplify all notions on a natural counterpart of the MaxSAT problem,

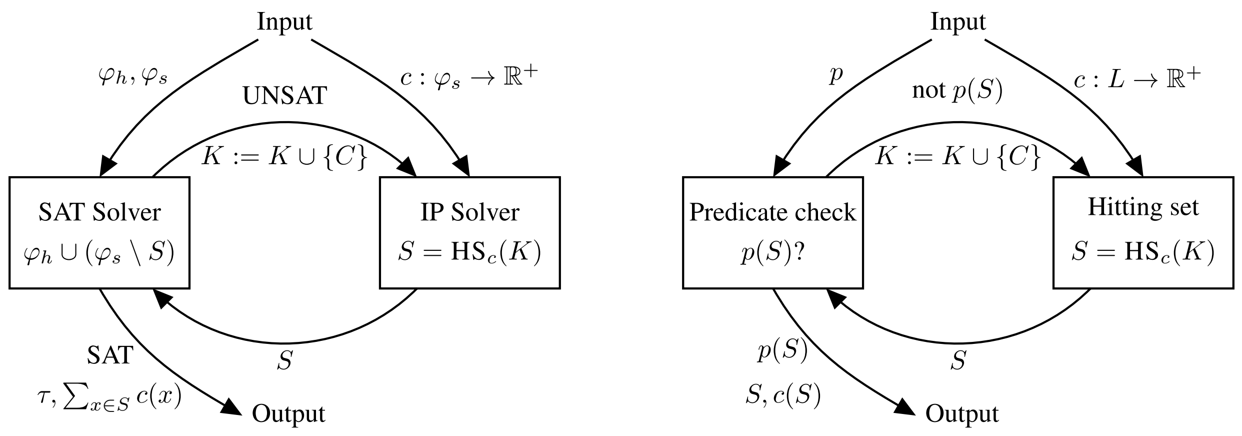

Figure 1: The MaxHS algorithm (left) and the generic framework for implicit hitting set algorithms (right).

namely, minimum satisfiability (MinSAT). In Partial MinSAT (or simply MinSAT) instance $\varphi = (\varphi_h, \varphi_s, c)$, solutions are the same as for MaxSAT. However, cost is incurred for *satisfying* clauses. In other words, for Partial MinSAT, the cost of a solution τ to φ is $\sum_{x \in \varphi_s} c(x) \cdot \tau(x)$.

Example 1. *Let $\varphi = (\varphi_h, \varphi_s, c)$ be an instance of MinSAT. We translate φ into our general setting by defining the minimization problem $P_{min} = (p_{min}, \varphi_s, c)$ so that predicate $p_{min}(S)$ holds for $S \subseteq \varphi_s$ iff $\varphi_h \wedge \bigwedge_{x \in \varphi_s \setminus S} \neg x$ is satisfiable, i.e., there is an assignment that satisfies the hard clauses and does not satisfy the soft clauses in $\varphi_s \setminus S$. For each minimum-cost solution $S \in Sol_c(P_{min})$ there is an optimal MinSAT solution τ to φ s.t. $c(S) = c(\tau)$ and vice versa.*

We note that there is a slight difference in the solutions to MinSAT and the solutions to the corresponding minimization problems as shown in the previous example: the former has satisfying assignments as solutions, while the latter has subsets of soft clauses as solutions. This is, however, not a real obstacle, since by implementing the predicate checks using SAT solvers, the SAT solver will also provide the corresponding truth assignment as a witness.

The next crucial definition lifts the unsatisfiable cores used in MaxHS to our general setting.

Definition 2. *Let $P = (p, L, c)$ be minimization problem. A set $C \subseteq L$ is a core of P if $C \cap S \neq \emptyset$ for all $S \in Sol(P)$.*

In words, a core C for a minimization problem P is a subset of L for which every solution contains an element in C. We denote the set of all cores of a minimization problem P by $Cores(P)$.

Example 2. *Consider again the minimization problem P_{min} from Example 1. A core $C \in Cores(P_{min})$ implies that every solution $S \in Sol(P_{min})$ contains a soft clause from C. If $C \in Cores(P_{min})$, then $\varphi_h \to \bigvee_{x \in C} x$ is a tautology, i.e., every assignment that satisfies the hard clauses also satisfies at least one clause of C (i.e., no assignment falsifies all clauses in C). Compared to unsatisfiable cores utilized in MaxHS, these are in a sense "tautological cores".*

A General Framework

Equipped with the general concept of cores for minimization problems, we now describe the general implicit hitting set algorithm. Pseudocode for the general setting is presented as Algorithm 1, and the general flow of the algorithm is illustrated in Figure 1 (right). The minimum-cost hitting set problem is now defined over a given cost function and the set of (general) cores. The "satisfiability" part consists here of a predicate check (Line 4) and, if this check fails to verify that a candidate is an actual solution, core extraction (Line 5). The basic idea of the algorithm is to guide the search by maintaining a set of computed cores K of the given problem and iteratively generating minimum-cost hitting sets of K. The algorithm starts with the empty set of cores and asks in each iteration for a minimum-cost hitting set S of the current set of cores K (Line 3). We then check whether S is a solution of P (Line 4). If $S \in Sol(P)$, we have found a minimum-cost solution. Otherwise, we extract a core based on the fact that S is not a solution and add the fresh core to K (Line 5). In general, one can define $extractcore(S) = L \setminus S$; we give more refined procedures for $extractcore$ later on. We repeat the process until a solution is reached, and terminate. The check for the *empty core* is a special case where no solution exists, i.e., $Sol(P) = \emptyset$. If $\emptyset$ is a core, then all solutions would have to intersect with it, which is not possible. This is accounted for by the test for empty cores on Line 2 which returns "no solution" in this case (Line 6).

Algorithm 1 Implicit Hitting Set Algorithm

```
Require: $P = (p, L, c)$ a minimization problem
Ensure: returns $S \in Sol_c(P)$ if $Sol(P) \neq \emptyset$, and otherwise "no solution"
1: $K := \emptyset$;
2: while $\emptyset \notin K$ do
3:     let $S$ be s.t. $S \in \mathrm{HS}_c(K)$;
4:     if $p(S)$ then return $S$;
5:     $K := K \cup \{extractcore(S)\}$;
6: return "no solution";
```

Example 3. *Consider P_{min} from Example 1. Instantiating Algorithm 1 for MinSAT, P_{min} results in a first iteration of checking whether $\emptyset$ is a solution (the minimum-cost candidate that is a hitting set of the empty set). If $\emptyset$ is not a solution, i.e., each assignment satisfying each hard clause satisfies at least one soft clause, it follows that $C = \varphi_s$ is a core of P_{max}, i.e., $\varphi_h \rightarrow \bigvee_{x\in\varphi_s}$ is a tautology. In subsequent iterations, minimum-cost hitting sets are generated from the current set of cores.*

We proceed with formal statements about the correctness of Algorithm 1. We first show that no solution is "missed" at any given iteration of the algorithm, i.e., no candidate with smaller cost than the current can be a solution.

Lemma 1. *Let $P = (p, L, c)$ be a minimization problem and $K \subseteq Cores(P)$ a set of cores for P. If $S = \mathrm{HS}_c(K)$, then for all S' with $S' \subset S$ or $c(S') < c(S)$ it holds that $S' \notin Sol(P)$.*

Next, if a candidate S is not a solution, then $L \setminus S$ is a core of P and is not contained in previously found cores.

Lemma 2. *Let $P = (p, L, c)$ be a minimization problem and $K \subseteq Cores(P)$ a set of cores for P. If $S = \mathrm{HS}_c(K)$ and $S \notin Sol(P)$, then $(L \setminus S) \in Cores(P)$ and for any $C \subseteq (L \setminus S)$ with $C \in Cores(P)$ we have $C \notin K$.*

Proof. First, by Lemma 1, $S' \in Sol(P)$ does not hold for any $S' \subset S$. Thus $T \cap (L \setminus S) \neq \emptyset$ for all $T \in Sol(P)$, which implies that $(L \setminus S)$ is a core of P.

Now assume $C \subseteq (L \setminus S)$ is a core of P. Suppose $C \in K$. It follows that S does not hit C, which is a contradiction to S being a hitting set of K. Thus $C \notin K$. $\square$

Thus we can define $extractcore(S) = (L \setminus S)$ to assign a trivial core at each iteration. In practice, implicit hitting set algorithms can compute smaller cores, i.e., $C \subsetneq (L \setminus S)$. The only requirement is that C is a core of the problem.

Example 4. *Consider again P_{min} from Example 1 and let S be a candidate that is not a solution. An alternative $extractcore(S)$ procedure would be to minimize the unsatisfiable core. Let $C = \varphi_s \setminus S$. It holds that $C \in Cores(P_{min})$. This means $\varphi_h \rightarrow \bigvee_{c\in C} c$ is a tautology. One can generate a subset-minimal core by searching for a cardinality-minimal set $C' \subset C$ s.t. $\varphi_h \rightarrow \bigvee_{c\in C'} c$ is a tautology. Then $C' \in Cores(P_{min})$.*

Finally, towards correctness, we show termination assuming that HS and $extractcore$ are terminating subcalls.

Corollary 3. *Let $P = (p, L, c)$ be a minimization problem and $S \subseteq L$. If $extractcore(S)$ returns a core C of P with $C \subseteq (L \setminus S)$, then Algorithm 1 terminates.*

The next proposition builds upon the preceding results to establish correctness of Algorithm 1 given that $extractcore$ provides a means to extract cores.

Proposition 4. *Let $P = (p, L, c)$ be a minimization problem and $S \subseteq L$. Assume $extractcore(S)$ returns a core C of P with $C \subseteq (L \setminus S)$. It holds that Algorithm 1 returns $S \in Sol_c(P)$ if $Sol(P) \neq \emptyset$, and otherwise "no solution".*

Note that concrete instantiations of the general procedures mainly require an implementation for verifying $p(S)$, $extractcore$, and searching for minimum-cost hitting sets. In Section 4 we show how the predicate check and core extraction for the propositional abduction problem can be implemented with SAT solvers. The minimum-cost hitting set can naturally be computed by an IP solver.

Properties of the General Framework

In this section we give basic properties of the general framework, with an emphasis on the hitting-set relations between solutions and cores of a problem.[1] We start with a simple observation: each solution of a problem is a hitting set of some set of cores for the problem.

Lemma 5. *Let $P = (p, L, c)$ be a minimization problem and $K \subseteq Cores(P)$. Every $S \in Sol(P)$ is a hitting set of K.*

In the following proposition we summarize the relations between hitting sets of all cores for a problem and its solutions. In particular, a useful property is that minimum-cost (subset-minimal) hitting sets of all cores of a minimization problem P are minimum-cost (subset-minimal) solutions of P. The subset-minimal hitting sets and solutions are $\mathrm{HS}_{\subseteq}(K) = \{S \in \mathrm{HS}(K) \mid \nexists S' \in \mathrm{HS}(K) \text{ s.t. } S' \subset S\}$ and $Sol_{\subseteq}(P) = \{S \in Sol(P) \mid \nexists S' \in Sol(P) \text{ s.t. } S' \subset S\}$, respectively.

Proposition 6. *Let $P = (p, L, c)$ be a minimization problem and $K = Cores(P)$ the set of all cores of P. It holds that*

1. $\mathrm{HS}(K) \supseteq Sol(P)$*;*
2. $\mathrm{HS}_{\subseteq}(K) = Sol_{\subseteq}(P)$*;*
3. $\mathrm{HS}_c(K) = Sol_c(P)$*; and*
4. $S \in \mathrm{HS}(K)$ *iff* $\exists S'$ *with* $S' \subseteq S$ *s.t.* $S' \in Sol_{\subseteq}(P)$*.*

In general, $\mathrm{HS}(K) = Sol(P)$ does not hold, as shown in the following example.

Example 5. *Consider $P = (p, L, c)$ with $L = \{a, b\}$, $c(a) = c(b) = 1$, and predicate p defined by $p(\{a\})$ and $p(\{b\})$, i.e., $Sol(P) = \{\{a\}, \{b\}\}$. We have $Cores(P) = \{\{a, b\}\}$. Then $X = \{a, b\}$ hits the core in $Cores(P)$, but $X \notin Sol(P)$. Note that $X \notin \mathrm{HS}_{\subseteq}(Cores(P))$ and $X \notin \mathrm{HS}_c(Cores(P))$.*

Restricting to monotone predicates results in a 1-to-1 correspondence between hitting sets of all cores and solutions.

Corollary 7. *Let $P = (p, L, c)$ be a minimization problem and $K = Cores(P)$. If p is $\subseteq$-monotone, then $\mathrm{HS}(K) = Sol(P)$.*

Example 6. *Regarding P_{min} from Example 1, the predicate p_{min} is $\subseteq$-monotone.*

In the following we consider the question of which sets can be cores of a problem. The next lemma follows immediately from the definition of a core.

Lemma 8. *Let $P = (p, L, c)$ be a minimization problem.*

[1] All presented results hold also for maximization problems via a dual definition for cores: a core $C \subseteq L$ for a maximization problem P' is such that $C \cap (L \setminus S) \neq \emptyset$ for all $S \in Sol(P')$.

- *If* $C \in \mathit{Cores}(P)$ *then* $C' \in \mathit{Cores}(P)\ \forall C' \supseteq C$.
- *If* $S \in \mathit{Sol}(P)$ *then* $\nexists C \in \mathit{Cores}(P)$ *with* $C \subseteq (L \setminus S)$.

Let $\mathcal{P}_L$ be the set of all minimization problems over a domain L. Let $\mathcal{K}_L = \{\mathit{Cores}(P) \mid P \in \mathcal{P}_L\}$ be the set of all sets of cores of minimization problems over the set L. We now show that a set $K \subseteq 2^L$ is a core of a minimization problem $P = (p, L, c)$ iff K is upward-closed w.r.t. $\subseteq$, i.e., $C \in K$ implies $C' \in K$ with $C \subseteq C'$.

Proposition 9. *Let L be a set. It holds that $\mathcal{K}_L = \{K \subseteq 2^L \mid C \in K$ implies $C' \in K$ with $C \subseteq C'\}$.*

Solution Enumeration

Finally, we discuss how the general framework can be naturally extended to cover solution enumeration problems.

Enumerating minimum-cost solutions Let $P = (p, L, c)$ be a minimization problem. Given an $S \in \mathit{Sol}_c(P)$, for deriving a new minimum-cost solution, we add $C = L \setminus S$ to K and repeat the algorithm. Note that C is technically not a core of P, but a core of P' with changed predicate p' such that for any $X \in 2^L \setminus \{S\}$ we have $p(X)$ iff $p'(X)$ and $p'(S)$ does not hold. The previously computed cores in K remain cores for the new problem. Computation terminates when a hitting set S' with $c(S) < c(S')$ is derived.

Enumerating subset-minimal solutions Let $P = (p, L, c)$ be a minimization problem. If we want to compute all subset-minimal solutions $\mathit{Sol}_{\subseteq}(P)$, we search for an initial solution S with minimum cardinality (which is also subset-minimal). We then adopt a variant of the hitting set problem, in which additional constraints are allowed for *excluding* an element of a set X. We impose such a constraint on S, which ensures that no candidate will be a superset of S. The termination criterion is now that there is no solution to the modified hitting set problem.

4 Case Study: Propositional Abduction

We now instantiate the generic implicit hitting set framework (Algorithm 1) for propositional abduction. We start with formal preliminaries and, additionally, give a novel complexity result for abduction which suggests that harnessing the power of the implicit hitting set framework for the problem is a reasonable approach.

Propositional Abduction

A propositional abduction problem (PAP) instance consists of a set M of manifestations (or observations) which we are to explain by a subset of a set H of hypotheses. Such a set of hypotheses, together with a background theory T, is a solution to the given abduction problem if (i) it entails the manifestations and (ii) is consistent. Further, we are here interested in *minimum-cost* explanations under a cost function over the subsets of H.

Formally, a PAP is a quintuple $P = (V, H, M, T, c)$ with V a finite set of variables, H, M, and T are formulas over V, and $c : H \to \mathbb{R}^+$ a cost function. For $S \subseteq H$, we have $c(S) = \sum_{s \in S} s$. We define abd^P_S as a shorthand for $T \wedge \bigwedge_{s \in S} s$.

Definition 3. *Let $P = (V, H, M, T, c)$ be a PAP. The set of explanations of P is given by $\mathit{Expl}(P) = \{S \subseteq H \mid \mathit{abd}^P_S \not\models \bot, \mathit{abd}^P_S \models \bigwedge_{m \in M} M\}$. The minimum-cost solutions of P are $\mathit{Expl}_c(P) = \underset{E \in \mathit{Expl}(P)}{\arg\min}\ (c(E))$.*

Example 7. *Consider the theory $T = (r \wedge p \to m) \wedge (r \wedge q \to m)$ stating that r together with p or q explains the manifestation $M = \{m\}$. The hypotheses are $H = \{p, q, r\}$. Possible explanations are $\{p, r\}$, $\{q, r\}$, and $\{p, q, r\}$. Augmenting the instance with costs $c(r) = c(p) = 1$ and $c(q) = 2$ means that $\{p, r\}$ is the (in this case unique) minimum-cost explanation.*

Computational Complexity of Abduction

As for the complexity of propositional abduction, we recall the most relevant results for our work from (Eiter and Gottlob 1995b). All hardness results hold even if H and M are restricted to propositional variables, i.e., $H, M \subseteq V$. A language is in DP iff it is the intersection of a language in NP and a language in coNP. A problem is in Σ^{P}_2 if it can be decided with a non-deterministic polynomial time algorithm with access to an NP oracle. A problem is in Δ^P_3 if it can be decided with a deterministic polynomial time algorithm that may access a Σ^{P}_2 oracle. If additionally the number of Σ^{P}_2 oracle calls is bounded by $O(\log\ n)$ for instance size n, then the problem is in Θ^{P}_3.

It is DP-complete to verify whether a given $S \subseteq H$ is an explanation for a PAP. Deciding whether there exists an explanation, i.e., whether $\mathit{Expl}(P) \neq \emptyset$, is Σ^{P}_2-complete. For a given $a \in H$ it is Δ^P_3-complete to decide the relevance problem, i.e., whether there is an explanation $S \in \mathit{Expl}_c(P)$ for a given PAP P s.t. $a \in S$. If the costs are polynomially bounded w.r.t. the input size of a PAP, then the complexity of the relevance problem drops to Θ^{P}_3.

We now show a novel complexity result for abduction. Using results and assumptions of fixed-parameter complexity theory, we prove that there is no algorithm solving the explanation existence problem that (i) can make a constant number of NP oracle calls, and (ii) may be exponential in the search space of explanations, i.e., in parameter $|H|$, but is only polynomially influenced by the size of the abduction instance. In the context of our general implicit hitting set framework, this means that we cannot, in general, expect to solve propositional abduction within a constant number of NP checks (e.g. predicate checks) even when we may traverse all subsets of hypotheses.

Towards the result, recall that FPT denotes the complexity class of parameterized problems for which there exists an algorithm that decides the problem in time $f(k) \cdot n^{\mathcal{O}(1)}$, where $f(\cdot)$ is an arbitrary computable function that only depends on the parameter k. A *parameterized reduction* of a parameterized problem Π to a parameterized problem Π' is an FPT-algorithm that transforms an instance (I, k) of Π to an instance (I', k') of Π' such that: (i) (I, k) is a yes-instance of Π if and only if (I', k') is a yes-instance of Π', and (ii) $k' = g(k)$, where $g(\cdot)$ is an arbitrary computable function that only depends on k. Hardness and completeness with respect to parameterized complexity classes is defined analogously to the concepts from classical complexity theory, using parameterized reductions. The class $\mathsf{FPT}^{\mathsf{NP}}[f(k)]$ was recently introduced by (de Haan and Szeider 2014b; 2014a; Endriss, de Haan, and Szeider 2015) and contains all problems that can be solved by an FPT-algorithm that can use $f(k)$ many calls to an NP oracle.

We now show that deciding whether there is an explanation for a given PAP is $\mathsf{FPT}^{\mathsf{NP}}[f(k)]$-complete if parameterized by $|H|$. This means that, under complexity-theoretic assumptions, there are no FPT reductions from this problem to SAT (or a constant number of NP-oracle calls).

Proposition 10. *Deciding whether there exists an explanation for a given PAP is $\mathsf{FPT}^{\mathsf{NP}}[f(k)]$-complete when parameterized by $|H|$.*

Proof. Let $P = (V, H, M, T, c)$ be a PAP. For membership, verify for each $S \in 2^H$ whether $S \in \mathit{Expl}(P)$. Verifying whether $S \in \mathit{Expl}(P)$ is in DP.

To establish hardness, we give a parameterized reduction from the $\mathsf{FPT}^{\mathsf{NP}}[f(k)]$-complete problem BH(LEVEL)-SAT (de Haan and Szeider 2014a). Recall first the definition of the unparameterized problem BH$_n$-SAT. An instance of this problem is $I = (\chi_1, \ldots, \chi_n)$, where each χ_i is a formula. For $n = 1$, I is a yes-instance iff χ_1 is satisfiable. For an odd $n \geq 3$, I is a yes-instance iff χ_n is satisfiable or $(\chi_1, \ldots, \chi_{n-1})$ is a yes-instance of BH$_{n-1}$-SAT. For an even $n \geq 2$, I is a yes-instance iff χ_n is unsatisfiable and $(\chi_1, \ldots, \chi_{n-1})$ is a yes-instance of BH$_{n-1}$-SAT. A problem instance of the parameterized problem BH(LEVEL)-SAT (de Haan and Szeider 2014a) is of the form $I = (\chi_1, \ldots, \chi_k)$ with parameter k. The task is to decide whether I is a yes-instance of BH$_k$-SAT.

Let $I = (\chi_1, \ldots, \chi_k)$ be an arbitrary instance of the parameterized BH(LEVEL)-SAT problem. W.l.o.g. we assume disjoint vocabularies for each pair of formulas in I. Define $r(\cdot)$ as a recursive function. If $k = 1$, then $r(I) = h_1$. If $k \geq 2$ is odd, then $r(I) = (h_k \vee (r(\chi_1, \ldots, \chi_{k-1})))$. If $k \geq 2$ is even, then $r(I) = (\neg h_k \wedge (r(\chi_1, \ldots, \chi_{k-1})))$. Define $T = (r(I) \rightarrow q) \wedge \bigwedge_{1 \leq i \leq k} (h_i \rightarrow \chi_i)$, $M = \{q\}$, and $V = \mathit{var}(T) \cup H \cup M$. We show that there is an explanation iff I is a yes-instance. First, we observe that if τ is a satisfying assignment of T and χ_i is unsatisfiable, then $\tau(h_i) = 0$. Second, I is a yes-instance if τ' satisfies $r(I)$ with $\tau'(h_i) = 1$ iff χ_i is satisfiable.

Assume that I is a yes-instance. Then $E = \{h_i \mid \chi_i \text{ satisfiable}\}$ is an explanation of the constructed abduction instance, since every satisfying assignment of $T \wedge \bigwedge_{h_i \in E}$ also satisfies $r(I)$ and in turn q. It is immediate that $T \wedge \bigwedge_{h_i \in E}$ is satisfiable.

Assume that E is an explanation of the constructed abduction instance. Suppose I is not a yes-instance. Then τ' defined as above does not satisfy $r(I)$. It holds that $E \cap \{h_i \mid \chi_i \text{ unsatisfiable}\} = \emptyset$ due to $h_i \rightarrow \chi_i$ in theory T. Thus $E \subseteq \{h_i \mid \chi_i \text{ satisfiable}\}$. It holds that τ' is a satisfying assignment of $T \wedge \bigwedge_{h_i \in E}$. But then q is not entailed by $T \wedge \bigwedge_{h_i \in E}$, which is a contradiction with E being an explanation of the abduction instance. □

Instantiating the Implicit Hitting Set Framework

We instantiate the general framework of implicit hitting set algorithms for abduction. Finding a minimum-cost explanation for a PAP $P = (V, H, M, T, c)$ is seen as a minimization problem in our setting. We define $P_{abd} = (p_{abd}, H, c)$ as the corresponding minimization problem. Predicate $p_{abd}(S)$ for $S \subseteq H$ holds if S is an explanation for P. It holds that $\mathit{Sol}(P_{abd}) = \mathit{Expl}(P)$. For the instantiation of Algorithm 1, we need procedures for (i) checking if a set $S \subseteq H$ is an explanation, (ii) core extraction, and (iii) finding minimum-cost hitting sets. We will apply SAT solvers for (i) and (ii), and an IP solver for (iii).

Let $P = (V, H, M, T, c)$ be a PAP. The function IP(K) solves for $K \subseteq 2^H$ (with variables v_h for $h \in H$)

$$\text{MINIMIZE} \sum_{h \in H} c(h) \cdot v_h \tag{1}$$

$$\text{SUBJECT TO} \sum_{h \in C} v_h > 0 \text{ for all } C \in K, \tag{2}$$

$$v_h \in \{0, 1\} \text{ for all } h \in H, \tag{3}$$

and returns a solution as a set S (variables assigned 1).

Algorithm 2 shows the instantiation of Algorithm 1 for abduction. We illustrate the workflow in Figure 2. We begin with a simple abduction-specific addition, namely the SAT call in Line 2 that checks whether the whole set of hypotheses does not entail the manifestations.[2] We denote with (sat, τ) the result of a SAT-call, i.e., sat is true iff the formula is satisfiable and in this case τ is a satisfying assignment. If the corresponding formula is satisfiable, then no subset of hypotheses entails the manifestations, and thus there are no explanations. In the main while-loop, we first check if the current candidate set S entails the manifestations in Line 5. If not, we obtain a satisfying assignment τ based on which we extract the core $\{h \in H \mid \tau(h) = 0\}$, as S is not an explanation. This core encodes that every explanation must contain a hypothesis assigned to 0 by τ. For every $S' \subseteq \{h \in H \mid \tau(h) = 1\}$, i.e., every candidate that is a subset of hypotheses assigned to 1 by τ, it holds that τ satisfies $abd^P_{S'} \wedge \bigvee_{m \in M} \neg m$, implying that $S' \notin \mathit{Expl}(P)$. The second condition of explanation is checked next (Line 8). If abd^P_S is unsatisfiable, we know that S is not an explanation, and extract the trivial core. If both conditions for explanations hold for candidate S, we have found a minimum-cost explanation. If not, we generate the next candidate explanation by the IP call. In case $S = H$, we can return "no solution" since the first check on Line 2 already has taken care of the case that $S = H$ is not an explanation.

To prove correctness of Algorithm 2, it suffices to show that at each iteration of the main loop, if S is not a solution, then the new set added to K is a core of the PAP.

Lemma 11. *Let P be a PAP and $K \subseteq \mathit{Cores}(P_{abd})$ be a set of cores for P_{abd}. Further, let $S = $ IP(K). The following statements hold.*

1. *If abd^P_S is unsatisfiable, then $(H \setminus S) \in \mathit{Cores}(P_{abd})$; and*

[2]This is in analogy with first checking if the set of hard clauses of a MaxSAT instance is satisfiable.

Algorithm 2 AbHS

```
Require: PAP $P = (V, H, M, T, c)$
Ensure: returns $S \in Expl_c(P)$ if $Expl(P) \neq \emptyset$, and otherwise "no solution"
 1: $K := \emptyset$; $S := \emptyset$;
 2: $(sat, \tau) := \text{SAT}(abd_H^P \wedge \bigvee_{m \in M} \neg m)$;
 3: if $sat$ then return "no solution";
 4: while $S \neq H$ do
 5:     $(sat, \tau) := \text{SAT}(abd_S^P \wedge \bigvee_{m \in M} \neg m)$;
 6:     if $sat$ then $K := K \cup \{\{h \in H \mid \tau(h) = 0\}\}$;
 7:     else
 8:         $(sat, \tau) := \text{SAT}(abd_S^P)$;
 9:         if not $sat$ then $K := K \cup \{(H \setminus S)\}$;
10:         else return $S$;
11:     $S := \text{IP}(K)$;
12: return "no solution";
```

2. *if τ is a model of $abd_S^P \wedge \bigvee_{m \in M} \neg m$, then* $\{h \in H \mid \tau(h) = 0\} \in Cores(P_{abd})$.

Note that from Lemma 2 and Lemma 11 it follows that Algorithm 2 produces a fresh core in each iteration, or terminates. Overall termination and correctness now straightforwardly follows from Proposition 4.

Corollary 12. *Let P be a PAP. If $Expl(P) \neq \emptyset$ then Algorithm 2 returns an $S \in Expl_c(P)$, and otherwise returns with "no solution".*

Finally, we note that the IP formulation of the minimum-cost hitting set problem can be strengthened via a simple domain-specific observation. Namely, if abd_S^P is unsatisfiable, and hence S is not a solution (Lines 8–9 of Algorithm 2), then we also know that $abd_{S'}^P$ is unsatisfiable for *any* $S' \supset S$. Hence no superset of S can be an explanation. This allows for strengthening the IP formulation by replacing the constraint $\sum_{h \in H \setminus S} v_h > 0$ with the constraint $\sum_{h \in S} v_h < |S|$. As a result, S and all its supersets are ruled out from the set of solutions to the IP. Furthermore, since S was an optimal solution to the previous IP, we know that no subset of S can be an optimal solution to the IP. These together imply that all subsequent optimal solutions to the IP must be incomparable to S w.r.t. $\subseteq$. In the experiments reported on in the following, we consider both the "original" IP formulation and the strengthened version, referring to the algorithm using the strengthened IP formulation as AbHS+.

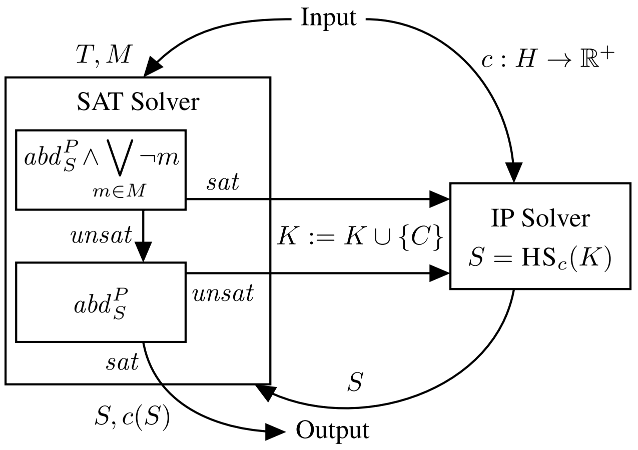

Figure 2: Instantiation of the general implicit hitting set framework for propositional abduction

5 Experiments

We proceed with an overview of empirical results on a prototype implementation of the instantiation of the general framework to propositional abduction.

We generated abduction instances based on crafted and industrial instances from MaxSAT Evaluation 2014 on which the running time of the state-of-the-art MaxSAT solver LMHS (http://www.cs.helsinki.fi/group/coreo/lmhs/) was at most five minutes. For a MaxSAT instance with hard clauses φ_h, soft clauses φ_s, and cost function c, we constructed an abduction instance $P = (V, H, M, T, c)$ as follows. Set the soft clauses $H = \varphi_s$ as the hypotheses, using the original weight, and the hard clauses as the theory, i.e., $T = \varphi_h$. For manifestations, compute the set M of literals entailed by the formula $\varphi_s' \wedge \varphi_h$ by the backbone solver minibones (Janota, Lynce, and Marques-Silva 2015), where φ_s' are the satisfied clauses of the optimal MaxSAT solution τ found by LMHS. Select at random a subset $M' \subseteq M$ for each size $|M| = k \in \{5, 10, 15\}$. (For nontrivial instances, we only considered entailed literals whose variables do not occur in soft clauses.) This construction ensures that the subset φ_s' is an explanation for P, but potentially not a minimum-cost one. This gave a total of 1641 abduction instances.

Our prototype implementation, AbHS, including the AbHS+ variant, employs MiniSAT 2.2.0 as the SAT solver, and CPLEX as the IP solver. We compare the performance of AbHS and AbHS+ to a natural encoding of propositional abduction as disjunctive logic programming under answer set semantics (ASP). Our current implementation of AbHS, the ASP encoding, and the benchmarks are made available at http://cs.helsinki.fi/group/coreo/abhs/.

We encode an abduction instance via facts in ASP and derive explanations via standard guess & check methodology, i.e., we guess an explanation candidate and verify (i) consistency by simple ASP constraints and (ii) entailment utilizing the so-called saturation technique requiring disjunctive rules (see (Eiter and Gottlob 1995a) for details on this technique). Finally, weak constraints ensure that optimal answer sets correspond to minimum-cost explanations.

For solving the ASP instances, we used the state-of-the-art disjunctive ASP solver Clingo 4.5.3 (Gebser et al. 2011). We tested both its default branch-and-bound based algorithm as well as the unsatisfiable core based algorithm the solver implements, invoked via the option `--opt-strategy=usc`. The experiments were run on 2.83-GHz Intel Xeon quad-core machines with 32-GB RAM under Debian Linux 8.0. A per-instance time limit of 1800 seconds was enforced.

Results are shown in Figure 3 for the default ASP approach (circle markers), the unsatisfiable core based ap-

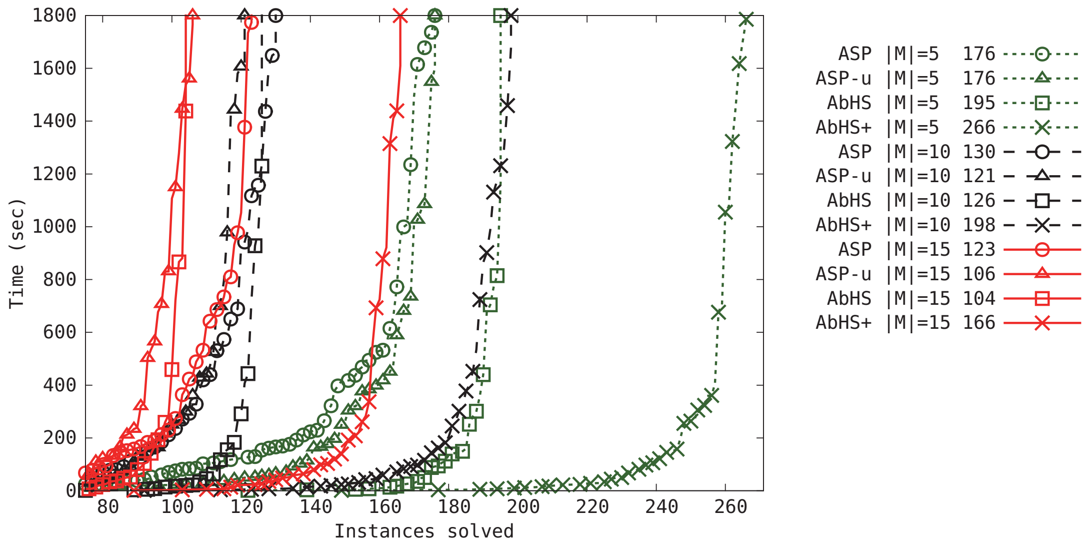

Figure 3: Comparison of AbHS and ASP.

proach ASP-u (triangle markers), AbHS (square markers), and AbHS+ (cross markers). Results are split by number of manifestations: $|M| = 5$ (dotted green lines), $|M| = 10$ (dashed black lines), or $|M| = 15$ (solid red lines). For each solver and $|M|$, the plot gives the number of instances (x-axis) for different *per-instance* time limits (y-axis). The numbers of instances solved under the 1800-second time limit are given in the plot key. Our base version AbHS is essentially already on par with the disjunctive ASP approach. The AbHS+ variant, however, clearly improves on the ASP approach, solving approximately 35% (for $|M| = 15$) to 50% (for $|M| = 5$) more instances than Clingo (both the default branch-and-bound version as well as the unsatisfiable core based version) on the disjunctive ASP encoding.

6 Related Work

We discuss connections to and differences with earlier work. Due to the generality of the approach, we will aim at briefly discussing these connections from various viewpoints. A fundamental ingredient in our proposal is that we aim at a *general* framework which allows for natural instantiations using SAT and IP solvers to problems *well beyond* NP. Furthermore, our instantiation to abduction yields to our best knowledge the first practical implementation to propositional abduction based on implicit hitting set algorithms.

The idea of implicit hitting set algorithms can be traced back to the classical work of Reiter (1987), who describes a domain-specific approach to diagnosis based on the concepts of conflict sets (cores) and hitting sets, mainly with theoretical motivations. A more general view on implicit hitting set algorithms was earlier proposed by Moreno-Centeno and Karp (2013), who focused on tackling NP-problems based on the idea of implicit hitting sets. In their proposal, the set of sets $\Gamma \subset 2^U$ to hit is given implicitly, and an abstract *separation oracle*, a polynomial time algorithm, certifies for a given $H \subseteq U$ whether H is a hitting set of the implicitly given set Γ to hit, or, otherwise, produces a set (called "circuit" in their approach) not hit by H. Their notion of circuits is covered by our notion of cores, and their separation oracle is covered by the predicate check and core extraction of our proposal. However, while our motivations are on beyond NP, Moreno-Centeno and Karp (2013) focused on problems in NP. Moreno-Centeno and Karp (2013) provide a heuristic instantiation of the idea to the NP-complete problem of multigenome alignment, without relying on SAT solvers, in contrast to our proposal.

In terms of other practical solvers arising from the proposed general framework, the MaxHS algorithm proposed for MaxSAT by Davies and Bacchus (2011; 2013a; 2013b), as discussed in this paper, provides key motivation for our work as a very successful approach to MaxSAT solving: most recently, the LMHS and MaxHS solvers, implementing these ideas, took top positions in the industrial and crafted categories of the 2015 MaxSAT Evaluation. In contrast to MaxHS, the more "typical" so-called *core-guided* approaches to MaxSAT solving (see, e.g., (Fu and Malik 2006; Heras, Morgado, and Marques-Silva 2011; Narodytska and Bacchus 2014; Martins et al. 2014; Morgado, Dodaro, and Marques-Silva 2014)) are driven by unsatisfiable cores provided by a SAT solver, without relying on a hybridization between SAT and IP solvers.

In terms of alternative general frameworks for tackling decision problems beyond NP, a central approach, often instantiated via SAT solvers, is counterexample-guided abstraction refinement (Clarke et al. 2003; Clarke, Gupta, and Strichman 2004), based on which practical solvers for various industrial and KR problems have been proposed (Janota et al. 2012; Janota and Marques-Silva 2011; Janota, Grigore, and Marques-Silva 2010; Wintersteiger, Hamadi, and de Moura 2010; de Moura, Ruess, and Sorea 2002; Barrett, Dill, and Stump 2002; Flanagan et al. 2003; Dvořák et al. 2014; Finkbeiner and Jacobs 2012).

Specifically for abduction, perhaps the closest algorithmic proposal to ours is the abstract approach by Satoh and Uno (2006) which, motivated by algorithms for the central itemset mining problem in data mining, iteratively collects subset-maximal *unexplanations*, i.e., sets that are not explanations, and constructs explanations by hitting the set of complements of the collected maximal unexplanations. The proposal of Satoh and Uno has not been implemented to the best of our knowledge, blocking a direct comparison.

There is a long line of research on other forms of abduction, including abductive logic programming (Kakas, Kowalski, and Toni 1992), the NP variants of cost-based abduction (see e.g. (Santos 1994)) and model-based diagnosis (Reiter 1987; Marques-Silva et al. 2015), abduction in system verification, e.g., abductive inference (Dillig and Dillig 2013), and abduction with first-order Horn theories (Ng and Mooney 1992). However, there is little work on systems for the actual propositional abduction problem focused on in this paper. This is why we provide a comparison with a native ASP-approach building upon the state-of-the-art disjunctive ASP solver Clingo.

The implicit hitting set approach for the NP-problem of minimum satisfiability (MinSAT) problem, outlined in this paper, has not been proposed earlier to the best of our knowledge. While not the main focus of this work, an interesting question is whether this approach could yield a competitive MinSAT solver, in analogy with the success of the MaxHS approach to MaxSAT. Further work on this would be motivated by the several recent works proposing different algorithmic approaches to MinSAT (Li et al. 2010; 2011; 2012; Zhu et al. 2012; Ansótegui et al. 2012; Kügel 2012; Heras et al. 2012; Ignatiev et al. 2013; Ignatiev, Morgado, and Marques-Silva 2014; Argelich et al. 2013).

Our $\mathsf{FPT}^{\mathsf{NP}}[f(k)]$-completeness result for abduction relies on the $\mathsf{FPT}^{\mathsf{NP}}[f(k)]$ class recently introduced by (de Haan and Szeider 2014b; 2014a; Endriss, de Haan, and Szeider 2015). Further related complexity results for abduction are shown in (Fellows et al. 2012; Pfandler, Rümmele, and Szeider 2013; Pfandler, Pichler, and Woltran 2015).

7 Conclusions

Motivated by the potential of lifting the successful MaxHS implicit hitting set approach for MaxSAT to problems well beyond NP, we outlined a general framework for implicit hitting set algorithms. Central to the proposed framework is that it allows for natural instantiations based on multiple SAT solvers and an optimization solver (for examples, an integer programming or a quadratic programming solver) for various important hard decision and optimization problems. As a case study, we detailed an instantiation of the framework for propositional abduction, a central KR problem that is hard for the second level of the polynomial hierarchy. The framework naturally allows for incorporating weight in the objective function, for example for weighted abduction. We showed that a prototype implementation of the instantiation outperforms a state-of-the-art disjunctive answer set solver on a natural encoding of propositional abduction. As the general framework allows for instantiations to various central problems in KR and AI, we see great potential in further instantiations of the framework for obtaining practically competitive solvers for other problem domains beyond NP.

Acknowledgements

This work has been funded by Academy of Finland through grants 251170 (COIN), 276412, and 284591, and the Doctoral Programme in Computer Science DoCS, University of Helsinki.

References

Ansótegui, C.; Li, C. M.; Manyà, F.; and Zhu, Z. 2012. A SAT-based approach to MinSAT. In *Proc. CAEPIA*, volume 248 of *Frontiers in Artificial Intelligence and Applications*, 185–189. IOS Press.

Argelich, J.; Li, C. M.; Manyà, F.; and Zhu, Z. 2013. MinSAT versus MaxSAT for optimization problems. In *Proc. CP*, volume 8124 of *Lecture Notes in Computer Science*, 133–142. Springer.

Argelich, J.; Li, C. M.; Manyà, F.; and Planes, J. 2015. MaxSAT Evaluation 2015. http://maxsat.ia.udl.cat/introduction/.

Barrett, C. W.; Dill, D. L.; and Stump, A. 2002. Checking satisfiability of first-order formulas by incremental translation to SAT. In *Proc. CAV*, volume 2404 of *Lecture Notes in Computer Science*, 236–249. Springer.

Brewka, G.; Eiter, T.; and Truszczyński, M. 2011. Answer Set Programming at a Glance. *C. ACM* 54(12):92–103.

Clarke, E. M.; Grumberg, O.; Jha, S.; Lu, Y.; and Veith, H. 2003. Counterexample-guided abstraction refinement for symbolic model checking. *J. ACM* 50(5):752–794.

Clarke, E. M.; Gupta, A.; and Strichman, O. 2004. SAT-based counterexample-guided abstraction refinement. *IEEE TCAD* 23(7):1113–1123.

Davies, J., and Bacchus, F. 2011. Solving MAXSAT by solving a sequence of simpler SAT instances. In *Proc. CP*, volume 6876 of *Lecture Notes in Computer Science*, 225–239. Springer.

Davies, J., and Bacchus, F. 2013a. Postponing optimization to speed up MAXSAT solving. In *Proc. CP*, volume 8124 of *Lecture Notes in Computer Science*, 247–262. Springer.

Davies, J., and Bacchus, F. 2013b. Exploiting the power of MIP solvers in MAXSAT. In *Proc. SAT*, volume 7962 of *Lecture Notes in Computer Science*, 166–181. Springer.

de Haan, R., and Szeider, S. 2014a. Compendium of parameterized problems at higher levels of the polynomial hierarchy. *Electronic Colloquium on Computational Complexity (ECCC)* 21:143.

de Haan, R., and Szeider, S. 2014b. Fixed-parameter tractable reductions to SAT. In *Proc. SAT*, volume 8561 of *Lecture Notes in Computer Science*, 85–102. Springer.

de Moura, L.; Ruess, H.; and Sorea, M. 2002. Lazy theorem proving for bounded model checking over infinite domains. In *Proc. CADE-18*, volume 2392 of *Lecture Notes in Computer Science*, 438–455. Springer.

Dillig, I., and Dillig, T. 2013. Explain: A tool for performing abductive inference. In *Proc. CAV*, volume 8044 of *Lecture Notes in Computer Science*, 684–689. Springer.

Dvořák, W.; Järvisalo, M.; Wallner, J. P.; and Woltran, S. 2014. Complexity-sensitive decision procedures for abstract argumentation. *Artif. Intell.* 206:53–78.

Eiter, T., and Gottlob, G. 1992. On the complexity of propositional knowledge base revision, updates, and counterfactuals. *Artif. Intell.* 57(2-3):227–270.

Eiter, T., and Gottlob, G. 1993. Propositional circumscription and extended closed-world reasoning are Π_2^P-complete. *Theoret. Comput. Sci.* 114(2):231–245.

Eiter, T., and Gottlob, G. 1995a. On the Computational Cost of Disjunctive Logic Programming: Propositional Case. *Ann. Math. Artif. Intell.* 15(3-4):289–323.

Eiter, T., and Gottlob, G. 1995b. The Complexity of Logic-Based Abduction. *J. ACM* 42(1):3–42.

Eiter, T., and Lukasiewicz, T. 2000. Default reasoning from conditional knowledge bases: Complexity and tractable cases. *Artif. Intell.* 124(2):169–241.

Endriss, U.; de Haan, R.; and Szeider, S. 2015. Parameterized complexity results for agenda safety in judgment aggregation. In *Proc. AAMAS*, 127–136. ACM.

Fellows, M. R.; Pfandler, A.; Rosamond, F. A.; and Rümmele, S. 2012. The parameterized complexity of abduction. In *Proc. AAAI*. AAAI Press.

Finkbeiner, B., and Jacobs, S. 2012. Lazy synthesis. In *Proc. VMCAI*, volume 7148 of *Lecture Notes in Computer Science*, 219–234. Springer.

Flanagan, C.; Joshi, R.; Ou, X.; and Saxe, J. B. 2003. Theorem proving using lazy proof explication. In *Proc. CAV*, volume 2725 of *Lecture Notes in Computer Science*, 355–367. Springer.

Fu, Z., and Malik, S. 2006. On solving the partial MaxSAT problem. In *Proc. SAT*, volume 4121 of *Lecture Notes in Computer Science*, 252–265. Springer.

Gebser, M.; Kaminski, R.; Kaufmann, B.; Ostrowski, M.; Schaub, T.; and Schneider, M. 2011. Potassco: The Potsdam answer set solving collection. *AI Commun.* 24(2):107–124.

Gottlob, G. 1992. Complexity results for nonmonotonic logics. *J. Logic Comput.* 2(3):397–425.

Heras, F.; Morgado, A.; Planes, J.; and Silva, J. P. M. 2012. Iterative SAT solving for minimum satisfiability. In *Proc. ICTAI*, 922–927. IEEE Computer Society.

Heras, F.; Morgado, A.; and Marques-Silva, J. 2011. Core-guided binary search algorithms for maximum satisfiability. In *Proc. AAAI*. AAAI Press.

Ignatiev, A.; Morgado, A.; Planes, J.; and Marques-Silva, J. 2013. Maximal falsifiability - definitions, algorithms, and applications. In *Proc. LPAR-19*, volume 8312 of *Lecture Notes in Computer Science*, 439–456. Springer.

Ignatiev, A.; Morgado, A.; and Marques-Silva, J. 2014. On reducing maximum independent set to minimum satisfiability. In *Proc. SAT*, volume 8561 of *Lecture Notes in Computer Science*, 103–120. Springer.

Janota, M., and Marques-Silva, J. P. 2011. Abstraction-based algorithm for 2QBF. In *Proc. SAT*, volume 6695 of *Lecture Notes in Computer Science*, 230–244. Springer.

Janota, M.; Klieber, W.; Marques-Silva, J.; and Clarke, E. M. 2012. Solving QBF with counterexample guided refinement. In *Proc. SAT*, volume 7317 of *Lecture Notes in Computer Science*, 114–128. Springer.

Janota, M.; Grigore, R.; and Marques-Silva, J. 2010. Counterexample guided abstraction refinement algorithm for propositional circumscription. In *Proc. JELIA*, volume 6341 of *Lecture Notes in Computer Science*, 195–207. Springer.

Janota, M.; Lynce, I.; and Marques-Silva, J. 2015. Algorithms for computing backbones of propositional formulae. *AI Commun.* 28(2):161–177.

Kakas, A. C.; Kowalski, R. A.; and Toni, F. 1992. Abductive logic programming. *J. Logic Comput.* 2(6):719–770.

Kügel, A. 2012. Natural Max-SAT encoding of Min-SAT. In *Proc. LION 6*, volume 7219 of *Lecture Notes in Computer Science*, 431–436. Springer.

Li, C. M.; Manyà, F.; Quan, Z.; and Zhu, Z. 2010. Exact MinSAT solving. In *Proc. SAT*, volume 6175 of *Lecture Notes in Computer Science*, 363–368. Springer.

Li, C. M.; Zhu, Z.; Manyà, F.; and Simon, L. 2011. Minimum satisfiability and its applications. In *Proc. IJCAI*, 605–610. IJCAI/AAAI Press.

Li, C. M.; Zhu, Z.; Manyà, F.; and Simon, L. 2012. Optimizing with minimum satisfiability. *Artif. Intell.* 190:32–44.

Marques-Silva, J.; Janota, M.; Ignatiev, A.; and Morgado, A. 2015. Efficient model based diagnosis with maximum satisfiability. In *Proc. IJCAI*, 1966–1972. AAAI Press.

Martins, R.; Joshi, S.; Manquinho, V.; and Lynce, I. 2014. Incremental cardinality constraints for MaxSAT. In *Proc. CP*, volume 8656 of *Lecture Notes in Computer Science*, 531–548. Springer.

Moreno-Centeno, E., and Karp, R. M. 2013. The implicit hitting set approach to solve combinatorial optimization problems with an application to multigenome alignment. *Oper. Res.* 61(2):453–468.

Morgado, A.; Dodaro, C.; and Marques-Silva, J. 2014. Core-guided maxsat with soft cardinality constraints. In *Proc. CP*, volume 8656 of *Lecture Notes in Computer Science*, 564–573. Springer.

Narodytska, N., and Bacchus, F. 2014. Maximum satisfiability using core-guided MaxSAT resolution. In *Proc. AAAI*, 2717–2723. AAAI Press.

Ng, H. T., and Mooney, R. J. 1992. Abductive plan recognition and diagnosis: A comprehensive empirical evaluation. In *Proc. KR*, 499–508. Morgan Kaufmann.

Pfandler, A.; Pichler, R.; and Woltran, S. 2015. The complexity of handling minimal solutions in logic-based abduction. *J. Logic Comput.* 25(3):805–825.

Pfandler, A.; Rümmele, S.; and Szeider, S. 2013. Backdoors to Abduction. In *Proc. IJCAI*, 1046–1052. IJCAI/AAAI Press.

Reiter, R. 1987. A theory of diagnosis from first principles. *Artif. Intell.* 32(1):57–95.

Santos, E. J. 1994. A linear constraint satisfaction approach to cost-based abduction. *Artif. Intell.* 65(1):1–28.

Satoh, K., and Uno, T. 2006. Enumerating minimal explanations by minimal hitting set computation. In *Proc. KSEM*, volume 4092 of *Lecture Notes in Computer Science*, 354–365. Springer.

Stillman, J. 1990. It's not my default: The complexity of membership problems in restricted propositional default logics. In *Proc. AAAI*, 571–578. AAAI Press / The MIT Press.

Stillman, J. 1992. The complexity of propositional default logics. In *Proc. AAAI*, 794–799. AAAI Press / The MIT Press.

Wintersteiger, C. M.; Hamadi, Y.; and de Moura, L. 2010. Efficiently solving quantified bit-vector formulas. In *Proc. FMCAD*, 239–246. IEEE.

Zhu, Z.; Li, C. M.; Manyà, F.; and Argelich, J. 2012. A new encoding from MinSAT into MaxSAT. In *Proc. CP*, volume 7514 of *Lecture Notes in Computer Science*, 455–463. Springer.