Abduction as Satisfiability

Petr Homola

LoqTek
phomola@loqtek.com

Abstract. Abduction is reasoning in propositional or first-order logic that provides explanations for observations. We show how context-free parsing and automated planning can be formulated using abductive rules and present a fast prover based on propositional satisfiability. Furthermore we show how weighted abduction can be efficiently implemented in the prover.

Keywords: abduction, satisfiability, parsing, planning

1 Introduction

Abduction is a form of ampliative reasoning that provides explanations for observations given a background theory. While abductive reasoning has proven useful in many areas of artificial intelligence and computer science (e.g., discourse interpretation [11], plan ascription [1], diagnostic expert systems), the fact that even its propositonal version is NP-hard restricts its use with large ontologies. This paper presents an algorithm for finding abductive proofs by converting the logical representation into a satisfiability problem. In conjunction with a state-of-the-art SAT solver, the transformation gives us an efficient abduction-based prover.

Abduction can be formally defined as follows: Let T be a first-order theory and O a set of literals ("observations"). Given a set of abductive rules (in the form $antecedent \rightarrow consequent$), abduction is finding a set of literals H ("hypothesis") such that

$$T \cup H \models O \text{ and } T \cup H \not\models \bot \tag{1}$$

that is, $T \cup H$ entails O and is consistent.

Abduction can be combined we other forms of reasoning, such as deduction. Indeed, most ontologies consist of both deductive and abductive rules. Particularly useful is weighted abduction, a variant of probabilistic abduction that can be used to find optimal proofs. What "optimal" means depends on what is being reasoned about. In discourse analysis, for instance, the least specific proof maximizing redundancy is preferred [11]. In the domain of automated planning, the shortest proof (i.e., the shortest plan leading to the desired goal state) is considered the best, and so forth.

We show how context-free parsing with feature structures (Section 2) and STRIPS-like (goal-driven) automated planning (Section 3) can be formulated

by means of abductive rules and present a transformation of abductive problems into propositional satisfiability (Section 4). An extension of the procedure to weighted abduction is described in Section 5. In Section 6 grounding is discussed and we conclude in Section 7.

2 Context-free Parsing

It is well-known that parsing can be thought of as deduction in first-order logic. On the other hand, a context-free grammar can be conceived as a set of abductive rules and the input as observations. In this setting, finding a parse is proving that there is an abductive proof of the "observed" terminal symbols (the input) that bottoms out in the initial nonterminal symbol S. For example, in the rule

$$S \rightarrow NP \ VP \tag{2}$$

S is the antecedent while NP and VP constitute the consequent. We use Kowalski's [15] notation and encode the input as a graph whose edges are labelled with the terminal symbols whereas nodes represent "interword points". For example, rule (2) is written in the abductive framework as

$$S(x,z) \rightarrow NP(x,y) \wedge VP(y,z) \tag{3}$$

which in turn is an abbreviation for

$$\forall x,z.S(x,z) \rightarrow \exists y.NP(x,y) \wedge VP(y,z) \tag{4}$$

where x, y, and z are interword points.[1]

The sentence (i.e., the "observation")

$$\text{Time flies like an arrow} \tag{5}$$

can be represented as

$$\begin{array}{l} token(0,1,\mathsf{time}) \wedge token(1,2,\mathsf{flies}) \wedge token(2,3,\mathsf{like}) \wedge token(3,4,\mathsf{an}) \wedge \\ \wedge token(4,5,\mathsf{arrow}) \end{array} \tag{6}$$

The lexicon consists of rules whose consequent is a terminal symbol while the antecedent is the corresponding preterminal symbol. For example, the ambiguous word *flies* is represented by two lexical rules:

$$\begin{array}{l} N(x,y) \rightarrow token(x,y,\mathsf{flies}) \\ V(x,y) \rightarrow token(x,y,\mathsf{flies}) \end{array} \tag{7}$$

If we have an observation (input tokens) and a set of abductive rules (based on a lexicon and a context-free grammar), we can use abductive reasoning to find a parse (or parses if the input is ambiguous) based on the following principles:

[1] We could also add the constraint $x < y \wedge y < z$ but it is not needed if the observations are a translation of a correct graph.

1. Each literal in the proof must be observed or abduced.
2. Each observed or abduced literal must be explained by at most one rule.

Note that we do not require that each observation be explained. However, we can require that $S(0, e)$ (where e is the end node) be abduced, i.e., it must be present in the proof. This integrity constraint ensures that the abductive proof explains all observations and intermediate abduced literals unless there is no proof that bottoms out in $S(0, e)$, in which case the input is not well-formed according to the grammar.

In the abductive framework, functional constraints (such as f-structures in LFG [13, 4]) can be easily added to the rules. Using a representation similar to that of Wedekind [17], the antecedents of the lexical rules are augmented with literals that represent feature structures and the antecedents of the grammar-specific rules are enriched with literals that represent functional constraints (i.e., unification). The literals that represent nonterminal symbols will now have the form

$$X(x, y, f) \tag{8}$$

where f is a constant representing a feature structure. The integrity constraint $S(0, e)$ becomes

$$S(0, e, f_1) \vee S(0, e, f_2) \vee \cdots \vee S(0, e, f_n) \tag{9}$$

where $f_1, \ldots, f_n$ are all possible feature structures associated with the nonterminal symbol S spanning the whole sentence.

The (somewhat simplified) lexical rule for *arrow* augmented with a feature structure is

$$\begin{array}{l} N(x, y, f_n(x, y)) \wedge attr(f_n(x, y), \mathsf{pred}, \mathsf{arrow}) \wedge \\ \wedge attr(f_n(x, y), \mathsf{number}, \mathsf{sg}) \rightarrow token(x, y, \mathsf{arrow}) \end{array} \tag{10}$$

that is, the corresponding feature structure has two attributes and its identifier is the Skolem function $f_n(x, y)$ where f_n is unique for every variant of the same word:

$$\begin{bmatrix} \text{PRED} & \text{'arrow'} \\ \text{NUMBER} & \text{sg} \end{bmatrix} \tag{11}$$

The unification of feature structures can be represented by a reflexive, symmetric, and transitive predicate eq for which the following additional axioms hold ($\mathcal{C}$ is the set of constants, $\mathcal{F}$ is the set of feature structures, and $\mathcal{A}$ is the set of attributes):

$$\begin{array}{l} \forall f_1, f_2 \in \mathcal{F}, a \in \mathcal{A}, v_1, v_2 \in \mathcal{F} \cup \mathcal{C}, x, y \in \mathcal{C} \\ 1.\ x \neq y \rightarrow \neg eq(x, y) \\ 2.\ attr(f_1, a, v_1) \wedge attr(f_2, a, v_2) \wedge eq(f_1, f_2) \rightarrow eq(v_1, v_2) \\ 3.\ attr(f_1, a, v_1) \wedge attr(f_2, a, v_2) \wedge \neg eq(v_1, v_2) \rightarrow \neg eq(f_1, f_2) \\ 4.\ attr(f_1, a, v_1) \wedge eq(f_1, f_2) \rightarrow attr(f_2, a, v_1) \end{array} \tag{12}$$

that is, two distinct constants can never be unified and each $f \in \mathcal{F}$ can be thought of as a function from $\mathcal{A}$ to $\mathcal{F} \cup \mathcal{C}$.

An example is in order. Let us assume the we want to analyze the fragment *will go* using the rule

$$VP \rightarrow Aux\ V \tag{13}$$

that is, an auxiliary and a following verb constitute a verb phrase. The corresponding (simplified) feature structures of the words are

$$\begin{bmatrix} \textit{will}: & \\ \text{TENSE} & \text{fut} \end{bmatrix} \quad \begin{bmatrix} \textit{go}: & \\ \text{PRED} & \text{`go'} \end{bmatrix} \tag{14}$$

and we want the rule to unify them.[2] The following axiom will effect what we want:

$$VP(x, z, f_1) \wedge eq(f_1, f_2) \rightarrow Aux(x, y, f_1) \wedge V(y, z, f_2) \tag{15}$$

that is, we unify f_1 and f_2, which yields the expected feature structure

$$\begin{bmatrix} \text{PRED} & \text{`go'} \\ \text{TENSE} & \text{fut} \end{bmatrix} \tag{16}$$

When backchaining on axiom (15), we assume the antecedent, that is, Aux and V constitute a VP and their feature structures can be unified. We could also use *attrIn* (set membership) instead of *attr* to express multiple dependencies (adjuncts) with the same attribute, such as

$$N(x, z, f_2) \wedge attrIn(f_2, adj, f_1) \rightarrow A(x, y, f_1) \wedge N(y, z, f_2)$$

$$\begin{bmatrix} \text{PRED} & \text{`time'} \\ \text{NUMBER} & \text{pl} \\ \text{ADJ} & \left\{ [\text{“good”}], [\text{“old”}] \right\} \end{bmatrix} \tag{17}$$

for *good old times*. It is not hard to see that the *eq*-axioms represent the unification of feature structures used in unification grammars.[3]

Completeness and coherence [4] can be encoded using a special predication, say, *hasAttr*:

$$\forall f, x, y. attr(f, x, y) \rightarrow hasAttr(f, x) \tag{18}$$

[2] Note that the feature structure of the auxiliary has no PRED attribute, for it is synsemantic. The main (autosemantic) word carries the semantic information of the phrase.

[3] The axioms define transitive closures in the sense of [17].

The lexical rules rules can then specify subcategorization frames by asserting $hasAttr(f, gf)$ or $\neg hasAttr(f, gf)$ where f is a feature structure and gf a subcategorizable grammatical function.

We could also add rules producing first-order logical forms (by augmenting a unification grammar as suggested by Homola [12]). In general, any annotation expressible by means of first-order literals can be added to the antecedents of the abductive rules.

Note that the abductive rules for parsing are effectively propositional,[4] i.e., we can use grounding and the transformation described below in Section 4 to find all abductive proofs.

3 Goal-driven Planning

Solving a STRIPS-like planning problem [7] can be thought of as finding an abductive proof. Let us assume that we want to find a plan of length n given an initial state, a goal state, and a set of actions with preconditions and effects.

The goal state is represented as $G(n)$, that is, the goal predicate(s) are true at time n. The initial state is represented as $I(0)$, i.e., it is an integrity constraint at time 0 that enforces an abductive explanation of the goal state unless there is no plan of length n. Actions are converted into abductive rules. The antecedent consists of the preconditions and an action literal and the consequent consists of the effects. For example,

$$\begin{aligned} &agentAt(x,t) \wedge boxAt(x,t) \wedge action(\mathsf{moveBox}, x, y, t) \rightarrow \\ &\rightarrow agentAt(y, t+1) \wedge boxAt(y, t+1) \end{aligned} \tag{19}$$

means that if the agent and the box are at location x at time t, the box can be moved to location y. The consequent represents the effects at time $t + 1$.[5]

The frame axioms have the form

$$agentAt(x,t) \rightarrow agentAt(x, t+1) \tag{20}$$

The rules are effectively propositional since all variables in the antecedent except in the action literal appear in the consequent and the actions do not add any new variables.

The same principles used for parsing

1. Each literal in the proof must be observed or abduced.
2. Each observed or abduced literal must be explained by at most one rule.

[4] In all rules, all variables in the antecedent appear in the consequent.

[5] We can add hard constraints such as

$$action(\mathsf{moveBox}, x, y, t) \rightarrow location(x) \wedge location(y) \wedge x \neq y$$

to restrict the state space.

ensure that abductive reasoning finds all plans of length n.

Automated planning formulated in the abductive framework has a desirable property: The sequence of actions need not be linear, that is, parallel actions can be generated if one so wishes. On the other hand, we can add constraints that exclude undesired configurations, such as

$$\forall x, y, t. x \neq y \rightarrow \neg agentAt(x, t) \vee \neg agentAt(y, t) \tag{21}$$

or rule out nonlinear plans by allowing only one action at any time t. We can even have independent background theories about the modelled world interoperating with the plan, but this goes beyond the scope of this paper.

4 Abduction as Propositional Satisfiability

Let us assume that we have a set of ground literals (observations, integrity constraints, and possibly hard constraints) and a set of propositionalized abductive rules. In this section we show how the rules can be converted into a SAT problem whose valuations encode all abductive proofs of the observations that bottom out in the integrity constraints.

1. We add the observations and constraints to the clause set (as unit clauses).
2. We numerate the rules. We write the i-th rule as

   $$a_{i,1} \wedge \cdots \wedge a_{i,n} \rightarrow c_{i,1} \wedge \cdots \wedge c_{i,m} \tag{22}$$

   For every rule, we add

   $$r_i \rightarrow a_{i,1} \wedge \cdots \wedge a_{i,n} \wedge c_{i,1} \wedge \cdots \wedge c_{i,m} \tag{23}$$

   to the clause set (r_i is an auxiliary propositional variable representing the i-th rule). That is, if an abductive rule is applied, the literals in its consequent must be observed or abduced and the rule adds the literals in its antecedent to the proof.
3. For every literal which is not an observation or a known fact we add

   $$l \rightarrow \bigvee \{r : l \text{ appears in the antecent of } r\} \tag{24}$$

   to the clause set. That is, l must be licensed by at least one rule.
4. For every literal which is observed or abduced we add

   $$l \rightarrow \neg r_i \vee \neg r_j \tag{25}$$

   to the clause set where r_i and r_j ($i \neq j$) are all possible rule pairs that have l in the consequent. That is, l must be explained by at most one rule.

The clause set obtained in steps 1–4 can be evaluated for satisfiability by a propositional SAT solver (we use MiniSAT). It is not hard to see that there is a 1-to-1 correspondence between valuations of the clause set and abductive

proofs. The observations, constraints, used rules (r_i), and abduced literals are true. All other literals are false. In the case of parsing (Section 2), the parse can be decoded from the positive rule literals. In the case of planning (Section 3), the plan can be decoded from the abduced action literals.

A simple example is given in form of a graph in Figure 1. The observation is d and the abductive rules are

$$\begin{array}{l} a \wedge b \rightarrow d \\ b \wedge c \rightarrow d \end{array} \tag{26}$$

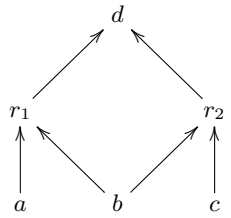

Fig. 1. Abductive graph

d appears in the proof because it is an observation. r_1 and r_2 exclude each other because they share a propositional variable in the consequent (i.e., $\neg r_1 \vee \neg r_2$). If r_1 is applied, a, b and d must appear in the proof. Likewise, if r_2 is applied, b, c and d must appear in the proof. If a appears in the proof, r_1 must be true because it is the only rule with a in the antecedent. If b appears in the proof, r_1 or r_2 must be true, and so forth. The clause set can be thought of as constraints that ensure that all its valuations conform to the rules of abductive reasoning. The subgraph obtained from the abductive graph by removing all false propositional variables (according to a valuation) represents an abductive proof that explains the observations and satisfies all constraints.

In the case of parsing, our algorithm seems to be faster in practice than chart-based algorithms (whose theoretical complexity is $O(n^3)$). However, the algorithm becomes much more interesting if we use it for parsing with unification grammars, which is known to be NP-hard [2] (although natural languages may well be parsable in subexponential time [6]). Note that the parser is neither top-down nor bottom-up since the SAT solver alone decides the order of steps carried out to find a valuation.

In the case of automated planning, the abductive approach does not give us a faster algorithm (it is very similar to SATPLAN [14]), but since the process of finding abductive proofs can be visualized by means of rule graphs, it reveals that SATPLAN and GRAPHPLAN [3] are merely two manifestations of the same principle.

5 Weighted Abduction

We are using SAT-based abduction in a project that includes natural language understanding. A unification grammar is used for parsing. Subsequently, Hobbs et al.'s [11] weighted abduction is used to interpret the logical form(s) in the discourse context at hand.

In weighted abduction, rules of the following form are used:

$$p_1^{\omega_1} \wedge \cdots \wedge p_n^{\omega_n} \rightarrow q \tag{27}$$

When backchaining on a consequent whose weight is ω, the weight of the (assumed) i-th literal in the antecedent is

$$\omega_i \omega \tag{28}$$

If $\sum_{i=1}^{n} \omega_i \omega$ is less than 1, proofs that explain the consequent by assuming some literals that make it true are preferred. If $\sum_{i=1}^{n} \omega_i \omega$ is greater than 1, it is cheaper to assume the consequent. The weights used in rules depend on the modelled problem. In the domain of language understanding, for example, in a rule such as

$$elephant(x)^{\omega_1} \rightarrow mammal(x) \tag{29}$$

we want $\omega_1 > 1$ for the class of individuals described by the antecedent is more specific than that described by the consequent.

Weighted abduction requires finding all abductive proofs and assigning a weight to them. If the used SAT solver returns at most one valuation, we can iteratively add its negation to the clause set until it is unsatisfiable. In other words, if $v_1, \ldots, v_n$ is a valuation, we add the clause

$$\neg(v_1 \wedge \cdots \wedge v_n) \equiv \neg v_1 \vee \cdots \vee \neg v_n \tag{30}$$

to the clause set.

A valuation obtained from the SAT solver can be used to assign a weight to the corresponding proof using a recursive algorithm. At the beginning, the observations have non-zero weights and the known facts have zero weights. An abduced literal without a weight must appear in at least one rule such that r_i is true. If there are literals in the consequent of the rule that do not have a weight, we recurse and compute those weights first. Otherwise, we compute the weight using the corresponding weight factor in the antecedent of the rule. If there are more applied rules, we use the lowest weight.

As for reasoning, we use a subset of the theory described in [10]. It seems that disjunction occurs only in the reified logical connectives (and', or', etc.) that operate on eventualities for which *Rexist* or $\neg$*Rexist* holds. Thus we use stable model semantics [8] to expand all disjunctions (most disjunctions are ruled out in the process of interpretation) and evaluate queries to the knowledge base(s)

using Chen and Warren's [5] SLG resolution, which seems to be very efficient in the domain of commonsense reasoning.[6]

Weighted abduction is considered intractable because in order to find the best proof we have to evaluate all possible proofs [1]. Possible solutions may be to extend the algorithm for use with a MAXSAT solver or to design a special SAT solver optimized for abductive tasks. Either way, there is a broad field for experiments and further research.

6 On Grounding

In the abductive framework, ontologies are typically expressed in first-order logic. In order to be able to use a SAT solver to find abductive proofs, we have to propositionalize (ground) the theory. This entails a few problems.

In order to find all valid proofs, the clause set (or the abductive graph) must contain all instantiations of the abductive rules that may occur in a proof. In the case of parsing or planning, all variables in the antecedent appear in the consequent (with the exception of the action predicates), which makes the grounding straightforward (in a top-down way starting with the observations). Problems arise if there is an existentially bound variable in the antecedent, as in the rules

$$\begin{array}{ll} & a.\ \forall x.[\exists y.infected(y, \mathsf{malaria}) \wedge bloodTransfused(y, x)] \rightarrow \\ & \quad \rightarrow infected(x, \mathsf{malaria}) \\ & b.\ \forall x.[\exists y.mosquito(y) \wedge infected(y, \mathsf{malaria}) \wedge bite(y, x)] \rightarrow \\ & \quad \rightarrow infected(x, \mathsf{malaria}) \end{array} \tag{31}$$

that is, if a person is infected with malaria, he may have undergone a blood transfusion from an infected donor or he may have been biten by an infected mosquito (in short, blood transfusions and mosquito bites transmit malaria). The problem is that when backchaining on axioms (31), the existence of y is either assumed or y is unified with a person or mosquito in the knowledge base. In the example at hand, the donor should be unified with an individual in the knowledge base (medical evidence) whereas in the case of a mosquito bite, the existence of an infected mosquito will virtually always be assumed. In general, we cannot exclude either possibility. Therefore, if there is a rule of the form

$$\forall x.[\exists y.P(y)] \rightarrow Q(x) \tag{32}$$

we add to the clause set all rule instantiations in which the individual substituted for y is compatible with P, and a special instantiation in which we substitute a new individual (a Skolem constant) for y. This solution to the grounding problem is particularly useful in the case of weighted abduction, for we can use weights to prefer one of the propositionalizations. In discourse interpretation, for instance, definite noun phrases (such as *the man*) should be unified with an individual

[6] Alternatively, Loveland's [16] resolution for "slightly non-Horn (near Horn) clause sets" could be used.

in the knowledge base while indefinite noun phrases (such as *a man*) are much more likely to introduce a new individual.

Even though every theory with the bounded-term-size property can be propositionalized, the number of rules can grow exponentially. It is therefore important to use a fast algorithm (in the typical case, grounding takes more time than SAT solving). We use a modified RETE algorithm to find all relevant rule instantiations.

7 Conclusions

We showed how context-free parsing and automated planning can be formulated in an abductive framework and how abductive proofs can be efficiently found by converting ontologies into a propositional clause set whose valuations correspond to abductive proofs. Since modern SAT solvers (such as MiniSAT) are very fast, the procedure presented in this paper can be used as an efficient abductive prover. Moreover, integrity constraints and hard (deductive) rules can be used to rule out proofs that are undesirable in the modelled domain.

Since abduction is widely used in many areas of artificial intelligence and computer science, such as discourse interpretation [11], plan ascription [1], diagnostic expert systems, etc., but is NP-hard, it is important to have a fast procedure for finding abductive proofs. We offer one by capitalizing on the enormous progress achieved in the field of SAT solving in the last decade.

The most important problem is to find a fast grounding algorithm for theories based on weighted abduction, as they may contain deductive rules, coercion rules [9], etc. Furthermore, we would like to test the performance of our algorithm with an ontology consisting of (tens of) thousands of facts and axioms.

Bibliography

[1] Appelt, D.E., Pollack, M.E.: Weighted Abduction for Plan Ascription. User Modeling and User-Adapted Interaction 2(1–2), 1–25 (1992)

[2] Berwick, R.: Computational Complexity and Lexical Functional Grammar. American Journal of Computational Linguistics 8, 20–23 (1982)

[3] Blum, A.L., Furst, M.L.: Fast Planning Through Planning Graph Analysis. Artificial Intelligence 90, 281–300 (1997)

[4] Bresnan, J.: Lexical-Functional Syntax. Blackwell Textbooks in Linguistics, New York (2001)

[5] Chen, W., Warren, D.S.: Tabled Evaluation with Delaying for General Logic Programs. Journal of the ACM 43(1), 20–74 (1996)

[6] Dalrymple, M., Kaplan, R.M., Maxwell III, J.T., Zaenen, A.: Formal Issues in Lexical-Functional Grammar. CSLI Publications (1995)

[7] Fikes, R., Nilsson, N.J.: STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving. Artificial Intelligence 2, 189–208 (1971)

[8] Gelfond, M., Lifschitz, V.: The Stable Semantics for Logic Programs. In: Kowalski, R., Bowen, K. (eds.) Proceedings of the 5th International Symposium on Logic Programming. pp. 1070–1080 (1988)

[9] Hobbs, J.R.: Syntax and Metonymy. In: Bouillon, P., Busa, F. (eds.) The Language of Word Meaning, pp. 290–311. Cambridge University Press (2001)

[10] Hobbs, J.R.: Encoding Commonsense Knowledge (2005), MS, Information Sciences Institute, University of Southern California, Marina del Rey

[11] Hobbs, J.R., Stickel, M., Appelt, D., Martin, P.: Interpretation as Abduction. Artificial Intelligence 63, 69–142 (1993)

[12] Homola, P.: Neo-Davidsonian Semantics in Lexicalized Grammars. In: Proceedings of the International Conference on Parsing Technologies. pp. 134–140 (2013)

[13] Kaplan, R.M., Bresnan, J.: Lexical-Functional Grammar: A formal system for grammatical representation. In: Bresnan, J. (ed.) Mental Representation of Grammatical Relations. MIT Press, Cambridge (1982)

[14] Kautz, H., Selman, B.: Planning as Satisfiability. In: Proceedings of ECAI-92. pp. 359–363 (1992)

[15] Kowalski, R.: Logic for Problem Solving. Oxford University Press (1979)

[16] Loveland, D.W.: Automated Theorem Proving: Mapping Logic into AI. In: Proceedings of the International Symposium on Methodologies for Intelligent Systems. pp. 214–229 (1986)

[17] Wedekind, J.: Classical Logics for Attribute-Value Languages. In: Proceedings of the 5th EACL conference. pp. 204–209 (1991)