Towards automated first-order abduction: the cut-based approach

MARCELO FINGER, *Department of Computer Science, Institute of Mathematics and Statistics, University of São Paulo, São Paulo, Brazil*
e-mail: mfinger@ime.usp.br

Abstract

Traditional abduction imposes as a precondition the restriction that the background information may not derive the goal data. In first-order logic such precondition is, in general, undecidable. To avoid such problem, we present a first-order cut-based abduction method, which has KE-tableaux as its underlying inference system. This inference system allows for the automation of non-analytic proofs in a tableau setting, which permits a generalization of traditional abduction that avoids the undecidable precondition problem. After demonstrating the correctness of the method, we show how this method can be dynamically iterated in a process that leads to the construction of non-analytic first-order proofs and, in some terminating cases, to refutations as well.

Keywords: Abduction, first-order abduction, cut-based abduction, tableaux, first-order tableaux.

1 A problem with first-order abduction

Abduction is traditionally understood as *inference to the best explanation* so it usually assumes that the goal data has not been yet explained. That is, the goal data to be explained is assumed not to follow from the background knowledge. While this may seem a natural assumption to trigger the abductive process, it ignores that it may not be decidable whether a given goal data is entailed by the background knowledge.[1]

This article is concerned with the inappropriateness of this assumption when abduction is performed having first-order logic as the underlying logic. Our goal is not only to philosophically and mathematically present abduction in a setting devoid of unfeasible preconditions, but also to derive practical consequences of this setting so as to allow one to *automate* the first-order abductive process in a computer.

The literature on automated abduction has been mostly concerned with how to compute an explanation, and how to define what a 'good' explanation is so as to guarantee that the 'best' explanation is computed. However, by posing the abductive problem in a generalized setting in which its effectiveness does not come into question, we are also widening the possibilities of defining 'good' explanations in a computationally efficient way. Furthermore, as will be seen, the techniques presented here may have applications in other areas beyond the search for explanations, such as automated proof construction and proof compression.

All these properties are obtained by performing *cut-based abduction*, a notion that was introduced by [12] in the propositional setting. In the present work, cut-based abduction will be developed into an algorithmic mechanism that abductively computes formulas using first-order logic as the underlying logic.

[1] The same criticism applies when the decidability of the underlying logic is *intractable*, although here the inadequacy of the assumption is even stronger.

Vol. 20 No. 2,
doi:10.1093/jigpal/jzq052 Advance Access published 17 December 2010

Given an abduction problem consisting of a background theory and a goal formula, cut-based abduction does not require that the entailment of the data by background theory be decided. As a result, cut-based abduction generalizes traditional abduction by covering simultaneously two cases:

- the goal data is not entailed by the background theory, which is the traditional abduction problem. In this case, an abductive hypothesis H is proposed as an acceptable explanation when the enriched theory obtained by adding the hypothesis to the background theory does entail the goal data.
- the goal data is entailed by the background theory. The generalized abductive task here is not explaining the goal data, but facilitating its proof. Now abduction provides a simpler proof of the goal data, where 'simpler' may mean 'shorter' or just 'easier to grasp' or a compromise between both.

Note that, as it is not known a priori in which case we are, the same abductive process has to work for both cases. In most cases, it will remain unknown *a posteriori* whether the goal formula follows from the background data or not. And here lies a novel property of the cut-based method, which is also responsible for its name.

It turns out that abductive steps may be combined by means of the *cut rule*, so as to compose a proof for the original sequent, or to obtain a counterexample for it. Therefore, the chaining of abductive steps leads to a decision procedure. In propositional logic it is guaranteed that such a process always terminates. In first-order logic the same iterated process can be performed, but of course no guarantee of termination is obtainable.

Chaining several abductive steps using the cut rule yields a non-analytic proof, which cannot be embedded in abductive processes based on analytic methods, such as [1, 18, 24]. Instead, we base the abductive method on an tableau inference method called KE-tableaux [11], which allows for the cut rule in the form of a *principle of bivalence*. In this case, one of the basic results is that each abductive step always terminates with a correct answer. The whole chain of abductive steps, however, may be infinite in first-order logic.

The method is considered a move 'towards' the automation of first-order abduction because it leads us to a situation where we are free to concentrate on computing the 'best' explanation, without concerns of undecidable preconditions. Apart from a comment on admissibility in Section 5, we do not concern ourselves with the selection of explanations among the set of explanations that the proposed algorithms compute.

The rest of the article is organized as follows. After presenting the notation, we review the literature of automated first-order abduction in Section 2, analysing strategies that deal with the computation of abductive formulas. In particular, the concepts underlying tableau-based abduction are presented in Section 3. The concepts involving cut-based abduction are presented in Section 4 and the algorithm for cut-based first-order abduction is presented and discussed in detail in Section 5. The unfolding of this method in a dynamic process is discussed in Section 6. A comparison of the proposed method with methods that combine unification and tableau methods for first-order logic lead to the suggestions of further work in Section 7.

1.1 Notation and initial definitions

We use letters p, q, r, s for atomic propositional or predicate symbols; an enumerable set of variables $\mathcal{V} = \{x_1, x_2, \ldots\}$; logical constants $\bot$ and $\top$; connectives $\wedge$ (conjunction), $\vee$ (disjunction), $\neg$ (negation) and $\rightarrow$ (implication); and quantifiers $\exists$ and $\forall$. Capital letters A, B, C, D, G, H represent formulas

and Greek capital letters Γ, Δ represent sets, multisets or sequences of formulas. A (first-order) sequent is a structure of the form $\Gamma \vdash \Delta$; in sequent context we write Γ, Δ for $\Gamma \cup \Delta$ and A for $\{A\}$.

In this notation, an a *abduction problem* is a pair $\langle \Gamma, G \rangle$, where Γ is the background theory and G is the goal data. The *traditional abductive assumption* states that the $\Gamma \nvdash G$ (and, in many cases, also that $\Gamma \nvdash \neg G$. The solution to an abduction problem is an *abduced formula* H such that

$$\Gamma, H \vdash G$$

a condition that is going to be maintained throughout this work as well. Of course, seeing abduction as an inference to the 'best' explanation imposes some extra conditions on the abduced formula, such as admissibility and minimality conditions. Here, we are going to be concerned with admissible conditions only. Usual admissibility conditions are:

- $H \nvdash G$: the abduced formula does not imply the goal data alone without taking into consideration the background theory; this condition precludes the trivial case where $H = G$.
- $\Gamma, H \vdash \bot$ only if $\Gamma \vdash \bot$: the abduced formula takes into consideration the goal data, instead of simply contradicting the background data; the case in which the background data is not taken into consideration in traditional abduction for it is blocked in the condition $\Gamma \not\models G$, but here we have to take this case into consideration as well.

These two conditions are, in general, undecidable in first-order logic, so the proposed method must provide means of satisfying those conditions by construction, without having to submit them to a decision procedure.

2 The Literature

Several proposals of automated abduction can be found in the literature. With respect to the undecidability of traditional abduction assumption in first-order logic, the literature has employed several strategies, among which:

- ignoring the problem;
- propositional and non-classical abduction;
- employing decidable fragments of first-order logic.

One of the first items in the literature to explicitly deal with abductive reasoning was de Kleer's Assumption-based Truth Maintenance Systems [13, 23], which dealt with abduction in propositional form. Formulas were manipulated in clausal form, which privileged resolution-based inference and the computation of prime implicants.

Poole's Default Reasoning [22] is a logical framework in which one is capable of generating explanations. The reasoning framework, called *Theorist*, employs quantified first-order logic with some syntactical limitation with regard the format of explanations and defaults. Notably, the framework does not mention any form of abduction assumption prior to explanation generation. However, the application of such framework to diagnosis employs only propositional formulas [21], indicating that when real automation was involved, only the propositional fragment was dealt with.

In the period in which logic programming dominated the Artificial Intelligence scene, abductive reasoning was preponderantly developed within the logic programming framework. One of the significant representatives of this kind of abductive system is Shanahan's view of 'prediction is deduction, explanation is abduction' [25], in which abduction is developed inside the representation framework of the Event Calculus, which employed logic programming with negation as failure.

In such a framework, universal quantification is dealt with implicitly so that all free variables occurring are implicitly universally quantified, and existential quantification is dealt with by a skolemization process prior to the performance of any type of inference. It is important to notice that, due to the use of negation-as-failure, the underlying logic is no longer first-order logic, as it has non-monotonic capabilities [14]. Several works in the intersection of abductive reasoning and logic programming were analysed in [16].

Outside the logic programming framework, the literature contains several works in which abduction is performed on decidable propositional logics:

- Carnielli describes a way in which abductive reasoning can be performed over paraconsistent propositional logics [7].
- Abductive reasoning was described over several decidable monomodal propositional logics [19]. In a similar way, the abduction can be performed over Description Logics [8]. Note that the modal and description logics that were employed in those works correspond to decidable fragments of first-order logic; other examples of abductive reasoning with a variety of underlying logics are described in [1].
- The class of n-Logics, containing first-order decidable fragments over finite domains, were employed as an approximation of effective first-order abductive reasoning [24]. Note that first-order logic over finite domains is still undecidable, so the problem of employing an n-Logic for decidable reasoning is that the size n of the domain has to be fixed a priori.
- A connection tableau-based method called SOLAR performs clausal *consequence finding*[20], seeking to generate theorems of interest. It distinguishes its goal from 'mere' theorem proving, which is seen a particular case of consequence finding. And it suggests that consequence finding can be used in abductive reasoning, for $\Gamma, H \models G$ can be reformulated as $\Gamma, \neg G \models \neg H$, that is, the negation of an abducted hypothesis can be seen as a 'theorem of interest' obtained from the background theory Γ enriched with the negation of the goal data to be explained.

An important class of abductive algorithms described in the literature is a collection of tableau-based abduction method. As they are relevant to the development of this work, we describe them in detail next.

3 Tableaux-based abduction

The notion of a tableau-based abduction was introduced by Mayer and Pirri [18] in a work that dealt with abduction via sequent calculi and tableaux. The paper dealt initially with classical propositional abduction and then generalized it to the first-order case. This was one of the first works to investigate automated abduction in a framework that did not involve resolution-based logic programming.

Resolution-based proof systems are not *analytic*, a fact that was criticized by the argument that analytic systems makes proof theoretical investigations clearer. Thus, Mayer and Pirri presentation of abduction chose cut-free sequent systems and Smullyan's analytic tableaux as proof-theoretical bases.

We present next their main ideas on how to perform abduction using analytic tableaux. However, we note that analyticity plays no role in the abduction process, a fact which we will return to in the presentation of cut-based abduction. What actually requires analytic proofs is the use of Smullyan's tableaux, which is based on the cut-free sequent calculus [26].

Mayer and Pirri restrict the search for first-order abduced formulas to the class of *C-formulas*, which they defined as the class of formulas made up from literals (atomic or negated atomic formulas)

TABLE 1. Analytic propositional expansion rules

$$\frac{\alpha}{\alpha_1 \quad \alpha_2} \qquad \frac{neg}{pos} \qquad \begin{array}{c} \beta \\ / \quad \backslash \\ \beta_1 \quad \beta_2 \end{array}$$

α	α_1	α_2
$TA \wedge B$	TA	TB
$FA \vee B$	FA	FB
$FA \rightarrow B$	TA	FB

neg	pos
$T\neg A$	FA
$F\neg A$	TA

β	β_1	β_2
$FA \wedge B$	FA	FB
$TA \vee B$	TA	TB
$TA \rightarrow B$	FA	TB

using only conjunction and quantifiers. Apparently, this restricted form of abduced formulas was deemed as an adequate format for an explanation, and it has no relation to the undecidability of the traditional abductive assumption. In fact, as the work on cut-based abduction has shown, the restriction to C-formulas can be eliminated.

The main idea of tableau-based abduction is to take an open tableau and, for each open branch, choose formulas that would make it close the branch if added to the initial tableau. By combining those formulas, one computes the abduced formula. In the propositional case, a simple conjunction of formulas is enough. In the first-order case, a special quantifier treatment is needed, which Mayer and Pirri called *reverse skolemization*.

We now describe the abduction procedure based on analytic tableau, as in [18]. We consider a tableau in which nodes are T- and F-signed formulas, such that TA and FA are called *conjugated* formulas. An *initial tableau* for a sequent $A_1, \ldots, A_n \vdash B$ has a single branch containing $TA_1, \ldots, TA_n, FB$; then we proceed by expanding the branches of the tableau by applying the expansion rules. Propositional expansion rules are of three types:

- α-Expansion: contains a premise α that, upon expansion, adds two formulas α_1 and α_2 to the branch.
- NEG-expansion: contains a negated premise *neg* that, upon expansion, adds a single *pos* formula to the branch.
- β-expansion: contains a premise β that, upon expansion, creates two branches with the current branch as a prefix and adding β_1 to one branch and β_2 to the other.

Table 1 presents the propositional analytic expansion rules of those three types.

A *branch* is thus a path from the root to the leaf. A branch of the tableau *closes* when it contains conjugated formulas TA and FA; the tableau *closes* when all its branches close, in which case we succeeded in proving the original sentence. If we use every possible tableau rule on a branch, we say it is *saturated*. A tableau is *completed* if every branch is either closed or open saturated. The idea of tableau theorem proving is to try to complete all branches; an open saturated branch guarantees the initial sequent is not provable and, in classical logic, it provides a countermodel to the original sequent. In propositional classical logic, all branches are finite, but in first-order logic some branches only saturate after an infinite number of expansions.

To deal with quantifier rules in first-order logic, we have to assume there is a countably infinite number of *parameters*, $\{a_1, a_2, \ldots\}$; parameters are constants not occurring in the initial tableau and correspond to Skolem constants used in the tableau expansion. The quantifier expansion rules are all linear, single-premised and provide a single conclusion, as illustrated in Table 2.

A γ-rule instantiates a γ-type formula with any term at all (which may even be a parameter occurring previously in the branch); unlike all other rules, a γ-type formula is not saturated on a branch by a single expansion, but by expanding it several times over all terms seen on that branch.

TABLE 2. Quantifier expansion rules

$$\frac{\gamma}{\gamma_1(t)} \quad \text{any term } t \qquad \frac{\delta}{\delta_1(a)} \quad \text{new parameter } a$$

γ	$\gamma_1(t)$
$T\forall xP(x)$	$TP(t)$
$F\exists xP(x)$	$FP(t)$

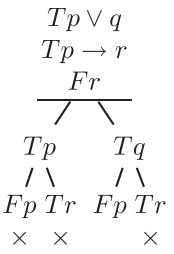
δ	$\delta_1(a)$
$T\exists xP(x)$	$TP(a)$
$F\forall xP(x)$	$FP(a)$

A δ-rule instantiates a δ-type formula with a single new parameter never occurring before in the tableau; in some cases, this may lead to an infinite number of expansions due to a combination of γ-type and δ-type expansions.

The abduction of C-formulas starts with a sequent that satisfies the traditional abduction assumption, that is, $\Gamma \nvdash G$. By the completeness of the tableau method, any completed tableau for it has at least one open saturated branch. For each such branch, choose a literal, and the abduced C-formula H, is the conjunction of the negation of all such literals. For the correctness of the abduction procedure, namely $\Gamma, H \vdash G$, it is required that if any parameter occurs in H it becomes a new variable universally quantified in H; this process constitutes the *reverse skolemization*[2] to be formally presented in Section 5.

EXAMPLE 1

Consider the invalid sequent $p \vee q, p \rightarrow r \nvdash r$ and a completed tableau for it.



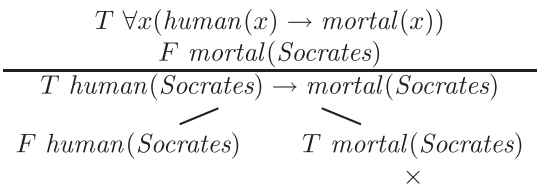
It has a single open saturated branch, with two candidate literals for the abduced explanation: Tq, which yields $H = \neg q$; and Fp, which yields $H = p$. It is easy to verify that both answers are correct.

EXAMPLE 2

Consider the invalid sequent involving quantifiers,

$$\forall x(\text{human}(x) \rightarrow \text{mortal}(x)) \nvdash \text{mortal}(\textit{Socrates})$$

and a completed tableau for it:



[2]In fact, this is a simplified version of Mayer's and Pirri's reverse skolemization, which will be sufficient for the current presentation.

In its single open saturated branch, we select F *human*(*Socrates*), which leads to $H = human(Socrates)$, as expected. We do not select F *mortal*(*Socrates*) as this leads to an abduced formula that derives G independently of the background knowledge, and is thus not admissible.

4 Cut-based abduction

Cut-based abduction, as presented in [12], is a generalization of traditional abduction, in which the decision of its initial assumption $\Gamma \vdash^{?} G$ is not required, so it simultaneously covers two cases:

(a) if $\Gamma \nvdash G$, the problem reduces to traditional abduction, which we call *hypothesis generation*;
(b) if $\Gamma \vdash G$ is provable, the task is not that of explaining a given set of data, but that of facilitating its proof, which we call *lemma generation*. We further expect the produced provable sequent $\Gamma, H \vdash G$ to have a simpler proof than $\Gamma \vdash G$. There is also a compromise to be reached between finding a simpler proof and finding a proof at a small computational cost.

Cut-based abduction expands the traditional *explanationist* view of abduction with a capacity of providing a simpler or more compact account of facts [15], thus presenting also a *simplificationist* view of abduction.

According to this latter view, the abduced hypothesis H can be seen as the computation of a *cut formula*. When the validity of $\Gamma \vdash G$ is not known, with a proof for $\Gamma, H \vdash G$ we can try to prove $\Gamma \vdash H, G$, as an 'auxiliary lemma'. If such proof is found, the two proofs are brought together via a cut inference, thus proving $\Gamma \vdash G$:

$$\dfrac{\dfrac{\Gamma \vdash H, G \quad \Gamma, H \vdash G}{\Gamma, \Gamma \vdash G, G}\,(Cut)}{\Gamma \vdash G}\,(Contraction)$$

It is known that proofs with non-analytic cuts can be exponentially smaller than cut-free and analytic proofs [4–6]. When lemma generation is successfully used to find shorter proofs we say we have *abduced proof efficiency*. We stress that the generalized abduction method developed here does not require any a priori knowledge of whether $\Gamma \vdash G$ or not.

As the cut rule plays a special role in this generalized view of abduction, so we do not use Smullyan's Analytic Tableaux as previous work, but employ instead the KE tableau method which is able to efficiently simulate sequent calculus with full use of cuts [11]. The cut-based abduction procedure is directly applied during a KE-tableau expansion, without prior knowledge of whether the input sequent is indeed provable or not. The basic idea remains that of Mayer and Pirri, with important differences, namely to generate for each open branch $\mathcal{B}$ of a KE-tableau (which needs not be a completed branch, that is, it may close when fully expanded) for a sequent $\Gamma \vdash G$, a sentence $H_{\mathcal{B}}$ whose addition to the background knowledge Γ is sufficient to close that particular branch. When it is not known a priori if a sequent is valid, we represent it as $\Gamma \vdash^{?} G$.

Another feature of this method is that, unlike [18], the format of the abduced formulas is not restricted to C-formulas. In fact, as there is no reason to suppose that C-formulas are preferable with regard to the class of all formulas, the set of admissible abduced formulas computed here is considerably larger. As the method is based on KE-tableaux, before we present the abduction algorithm, we discuss the principles underlying KE-tableaux.

TABLE 3. KE propositional expansion rules

α	neg	β
	pos	β_1^c
α_1		β_2
α_2		

FIG. 1. Principle of Bivalence

4.1 KE-Tableaux

KE-tableaux ([11], see also [9, 10]) were proposed by D'Agostino and Mondadori as a way of incorporating in a non-redundant way the cut rule into tableau proofs. This rule corresponds at the semantic level to the *principle of bivalence* (PB) which is essential for efficient proof-theoretical treatment of classical logic. In comparison with Smullyan's Analytic Tableau [26], KE-tableau have better computational properties; the use of the cut rule can be suitably restricted so as to preserve all desirable properties of cut-free proofs.

KE-tableaux also deal with signed formulas. We represent the conjugate of a signed formula S as S^c. Each connective is associated with a set of *linear expansion rules*; the rules for boolean connectives are shown in Table 3.

The α- and *NEG*-rules are exactly as in the rules for analytic tableau in Table 1, namely one-premised linear rules. However, the β rules do not branch the tableau and consist of two-premised linear rules, with a single conclusion. All KE linear expansion rules always have a *main premiss*, i.e. the one containing the connective to be eliminated; two-premiss rules also have an *auxiliary premiss*.

The *only* branching rule in KE is the PB, stating that a formula A must be either true or false, as illustrated in Figure 1, which corresponds to the cut rule.

The definitions of branch, tableau tree, closed branch, closed tableau, saturated branch and completed tableau are exactly as in the analytic case presented in Section 3. Any saturated branch provides a counter-model for the sequent.

Although the applications of PB may introduce arbitrary signed formulas, so violating the subformula property, D'Agostino and Mondadori [11] showed that there is no loss of completeness in restricting the applications of PB in such a way as to preserve the subformula property. Also, it is always possible to simulate an analytic deduction in KE with a linear increment of the number of symbols. On the other hand, there are some KE-proofs for which there only exist analytic proofs exponentially larger [11].

4.2 The cut-based abductive procedure

As in the previous case, abduction starts with a KE-tableau in which there is one or more open branches. Note that those branches need not be saturated, which poses an extra problem for the cut-based procedure, namely to decide when to stop the tableau expansion.

The generic computation of a hypothesis from a branch is as follows. Given a non-empty set of signed formulas $\Phi = \{TA_1, \ldots, TA_n, FB_1, \ldots, FB_m\}$, we compute the hypothesis $H(\Phi)$ as follows:

$$H(\Phi) = \begin{cases} \neg A_1 & , \text{ if } n = 1, m = 0 \\ B_1 & , \text{ if } n = 0, m = 1 \\ \neg(A_1 \wedge \ldots \wedge A_n) & , \text{ if } n > 1, m = 0 \\ B_1 \vee \ldots \vee B_m & , \text{ if } n = 0, m > 1 \\ (A_1 \wedge \ldots \wedge A_n) \rightarrow (B_1 \vee \ldots \vee B_m) & , \text{ if } n > 0, m > 0 \end{cases}$$

In [12], it was shown that given a KE-tableau branch containing formulas Ψ, and a non-empty set $\Phi \subseteq \Psi$, if we add $H(\Phi)$ as a top hypothesis, this branch can be expanded into a closed subtree; also, for each $\Phi \subseteq \Psi$ we have a candidate abduced formulas, not all of which will be admissible. If the initial tableau has more than one open branch, then we have to apply this method to each open branch $\mathcal{B}$ and take the conjunction of all $H_{\mathcal{B}}(\Phi)$, thus guaranteeing the correctness of propositional abduction.

EXAMPLE 3

We return now to the example in Example 1, with initial sequent $p \vee q, p \rightarrow r \nvdash r$ and a completed KE-tableau for it.

$$\begin{array}{c} Tp \vee q \\ Tp \rightarrow r \\ Fr \\ \hline Fp \\ Tq \end{array}$$

We first note that no branching was needed in this KE-expansion, illustrating its computational efficiency over the previous analytic proof, which could not be completed without branching.

We construct the set $\Psi = \{Fp, Tq, Fr\}$, and the following subsets and candidates:

Candidate	Φ	H
1	$\{Fp\}$	p
2	$\{Tq\}$	$\neg q$
~~3~~	~~$\{Fr\}$~~	~~r~~
4	$\{Fp, Tq\}$	$q \rightarrow p$
5	$\{Fp, Fr\}$	$p \vee r$
6	$\{Tq, Fr\}$	$q \rightarrow r$
7	$\{Tq, Fp, Fr\}$	$q \rightarrow (p \vee r)$

Candidate 3 was eliminated for it alone proves the goal data; while this in general is undecidable in first-order logic, it suffices to eliminate sets that only contain atoms occurring in the goal data.

Note that the admissible formulas generated by candidates 1 and 2 are the C-formulas generated in Example 1, while all the others are not C-formulas but perfectly admissible explanations for the goal data. Also note that all admissible abduced formulas logically imply the last one, which is called the *least compromising hypothesis* (LCH); this is the logically weakest formula to be computable by this method, although it may not be the preferred formula for some applications; in [12] it is shown that the LCH is *not* the best choice of abduced formula when searching for efficient cut formulas.

The discussion on cut-based abduction has so far concentrated on the propositional case; we will now turn to the first-order case.

5 The first-order algorithm

We start the discussion on first-order automated abduction by noting that the quantifier rules for KE-tableaux are exactly the same as those for the analytic tableaux, as presented in Table 2.

The quantifier rules are all linear, with a single premise and a single conclusion. However, a combination of γ- and δ-expansions has the peculiarity that it may lead to a γ-rule to be used an infinite number of times on a branch (see Example 5), so it never finitely saturates. This gives origin to infinite open branches, which lead to the impossibility of determining the termination of the tableau expansion in general for first-order tableaux.

The δ-rule, on the other hand, does not pose a problem for termination, but it does present a problem for abduction, as follows. Recall that the δ-rule inserts new *parameters* in the tableau, that is, it inserts new constants that have not occurred up to that point in the tableau expansion. The abductive reasoning using tableaux consists of adding enough information to the abduced hypothesis, so as to guarantee that the tableau closes. However, if such added information contains a parameter, the answer is *unsound*, as illustrated by the following example.

EXAMPLE 4

Consider the sequent, which is a variation of that in Example 2,

$$\forall x(\text{human}(x) \rightarrow \text{mortal}(x)) \vdash^{?} \forall y \text{mortal}(y)$$

and an incomplete tableau for it

$$\dfrac{\begin{array}{c} T\ \forall x(\text{human}(x) \rightarrow \text{mortal}(x)) \\ F\ \forall y \text{mortal}(y) \end{array}}{\begin{array}{ll} F\ \text{mortal}(a) & a \text{ is new} \\ T\ \text{human}(a) \rightarrow \text{mortal}(a) & \\ F\ \text{human}(a) & \end{array}}$$

The set of all admissible candidates for the single branch closing, is $\Phi = \{F\ \text{human}(a)\}$, which according to the abduction procedure leads to abducted hypothesis $H = \text{human}(a)$.

However, as can be easily verified, by making $H = \text{human}(a)$, the following sequent is invalid:

$$\forall x(\text{human}(x) \rightarrow \text{mortal}(x)), H \not\vdash \forall y \text{mortal}(y).$$

which means that this form of abduction is unsound; that is, the hypothesis added to the background information no longer derives the goal data.

The reason for this failure is the introduction of the parameter that has no meaning outside the tableau proof under expansion. In fact, that skolemization has to be reversed and transformed back into a quantified variable. For our method to be sound, all formulas entering the set of branch closing candidates have to undergo a reverse skolemization.

To perform abductive reasoning on the tableau expansion above, the only candidate formula is $F\ \text{human}(a)$, which by reversing the δ-expansion becomes $F\ \forall z \text{human}(z)$; in fact, $F\ \forall z \text{human}(z)$ is a δ-type formula whose δ-expansion on parameter a would lead to $F\ \text{human}(a)$. Now we can follow the original procedure and make $H = \forall z \text{human}(z)$, and the resulting sequent is clearly valid.

$$\forall x(\text{human}(x) \rightarrow \text{mortal}(x)), \forall z \text{human}(z) \vdash \forall y \text{mortal}(y).$$

Reverse skolemization is actually a recursive procedure on signed formulas, as a formula may contain more than one parameter, so the reverse skolemization depends on a list of parameters

π presented in the order they were introduced in the tableau, formally defined by:

$$\begin{array}{lcl} Skolem^{-1}(SA,\varnothing) & = & SA \\ Skolem^{-1}(SA(a),a\cup\pi) & = & S\mathcal{Q}zB(z) \\ & & \text{where } SB(a)=Skolem^{-1}(SA(a),\pi); \\ & & \mathcal{Q}=\begin{cases} \forall & \text{if } S=F \\ \exists & \text{if } S=T \end{cases}; \\ & & z \text{ is a variable not occurring in } B(a). \end{array}$$

Reverse skolemization was already employed by [18], but the version above is somewhat simpler in that it only introduces one level of quantification; the algorithm of [18] introduces several levels of alternate $\forall/\exists$ quantification, but it has to keep track of dependent quantifying variables.

EXAMPLE 5
Consider the sequent

$$\forall x\exists y m(y,x)\vdash^{?}\exists y\forall x m(y,x)$$

and a potentially infinite (thus incomplete) tableau for it

1.	$T\forall x\exists y m(y,x)$	*assumption*
2.	$F\exists y\forall x m(y,x)$	*assumption*
3.	$T\exists y m(y,t)$	γ on 1
4.	$F\forall x m(u,x)$	γ on 2
5.	$T m(a,t)$	δ on 3
6.	$F m(u,b)$	δ on 4
	$\vdots$	

Note that the γ-expansions on lines 3 and 4 cannot refer to parameters a and b (which would close that branch), as those were only introduced later on that branch expansion process. The terms u and t could be made identical (they are, in fact, forced to be identical in some tableau expanding algorithms), but this would not close the branch.

In fact, this tableau is a case of infinite failure. However, if its expansion is interrupted at the point displayed above, there are parameters a and b that have been introduced by δ-rules and there are also terms t and u introduced by γ-rules. Suppose we choose the largest closing set possible, namely $\Phi=\{Tm(a,t),Fm(u,b)\}$. By the propositional method described above,

$$H(\Phi)=Fm(a,t)\rightarrow m(u,b)$$

so when we apply reverse skolemization on that branch parameters $\{a,b\}$:

$$Skolem^{-1}(H(\Phi),\{a,b\})=F\forall zw(m(w,t)\rightarrow m(u,z))$$

which is a correct solution, as we can easily verify that a tableau for the sequent

$$\forall x\exists y m(y,x),\forall z\forall w(m(w,t)\rightarrow m(u,z))\vdash\exists y\forall x m(y,x)$$

closes. Note that the closing is obtained by considering terms t and u as constants, which can also be obtained if the γ-introduced terms are become existentially quantified variables in the abduced

hypothesis. That is, after reverse skolemization, the *existential closure* of the formula is taken to form abduced hypothesis. The following tableau also closes:

$$\forall x \exists y m(y,x), \exists t \exists u \forall z \forall w (m(w,t) \rightarrow m(u,z)) \vdash \exists y \forall x m(y,x)$$

In Example 5, the computation of the abductive hypothesis did not require a fully completed tableau, as the only tableau branch does not finitely saturate. So, in order to characterise the abduction procedure, one has to consider as input also a partially expanded tableau. In principle, any partially expanded tableau can be used for the abductive process.

LEMMA 1
Consider an open branch $\mathcal{B}$ in a partially expanded *KE* tableau $\mathcal{T}$ and a subset $\Phi \subset \mathcal{B}$ of its formulas such that each $SA \in \Phi$ has not been expanded. Then adding the complement of $Skolem^{-1}(H(\Phi))$ to the branch closes it.

PROOF. If no parameter occurs in $H(\Phi)$, then this is just the result of abduction correctness for quantifier-free tableaux obtained in [12].

So let $\pi = \{a_1, \ldots, a_k\}$ be the set of parameters occurring in $H(\Phi)$ and without loss of generality assume $H(\Phi)$ is of the form $FA(a_1, \ldots, a_k)$; if it is of the form TB, make it $F\neg A$. Then $Skolem^{-1}(H(\Phi)) = F\forall z_1 \ldots \forall z_k A(z_1, \ldots, z_k)$. By adding $T\forall z_1 \ldots \forall z_k A(z_1, \ldots, z_k)$ to $\mathcal{B}$ and applying k instances of the γ-expansion, each instantiating z_i with a_i, $1 \leq i \leq k$, we obtain the same formula one would obtain from Φ without reverse skolemization. So by the correctness of cut-based abduction in [12], the branch closes. ∎

Algorithm 5.1 presents the cut-based abductive algorithm for first-order logic.

Algorithm 5.1 Cut-based *FOLAbduction*$(\Gamma, G, \mathcal{T})$

Input: A sequent $\Gamma \vdash^? G$, and $\mathcal{T}$ a partially expanded tableau for it
Output: A hypothesis H such that $\Gamma, H \vdash G$

1: Let $\mathcal{B}_1, \ldots, \mathcal{B}_k$ be the open branches in $\mathcal{T}$, let π be the parameters used.
2: **for** $i = 1$ to k **do**
3: Let $\Psi_i = \{XA \in \mathcal{B}_i | XA \text{ unfulfilled in } \mathcal{B}_i\}$
4: Choose an admissible $\Phi_i \subseteq \Psi_i$
5: Let $H_i = H(\Phi_i)$
6: **end for**
7: **return** $Skolem^{-1}(H_1 \wedge \ldots \wedge H_k, \pi)$

Note that Algorithm 5.1 is a non-deterministic algorithm in two ways. One can choose the subset Φ_i of unfulfilled signed formulas to generate the abduced hypothesis. Furthermore, given a sequent $\Gamma \vdash^? G$ one can one expand the tableau in whatever order, and this expansion can halt at any time, which provides a further non-determinism to the abductive process. However, even with all this degree of non-determinacy, the correctness of the abduction process is guaranteed.

THEOREM 1 (Correctness of *FOLA* Algorithm)
The cut-based First-Order Logic Abduction Algorithm 5.1 is correct. That is, on input $\Gamma \vdash^? G$ it outputs a formula H such that $\Gamma, H \vdash G$.

PROOF. Consider any partially expanded KE-tableau $\mathcal{T}$ for $\Gamma \vdash^? G$. The previous result on the correctness of cut-based abduction in [12] guarantees the correctness of propositional abduction. Then

by applying Lemma 1 that deals with quantifiers, we have that adding to the tableau hypotheses the formula TH all open branches of $\mathcal{T}$ close, so by the soundness of first-order logic KE-tableau, $\Gamma, H \vdash G$. ∎

One important part of the algorithm above is that it does not determine how deep to expand the tableau at the point the abduction is executed. This is an important feature for applying abduction when it is not known a priori if the sequent is provable or not. This feature also provides a lot of flexibility to our method, allowing it to cope with 'hard' abduction problems.

EXAMPLE 6

Consider the following example, first proposed by Yamamoto [27] when showing the limitations of Inverse Entailment.

$$\text{even}(0), \forall x(\text{odd}(x) \rightarrow \text{even}(s(x))) \quad \vdash^{?} \quad \text{odd}(s(s(s(0))))$$

and the question is if the formula $\forall x(\text{even}(x) \rightarrow \text{odd}(s(x)))$ can be inferred. For that, we propose to apply a γ-expansion with $x = s(0)$ and then branch over the β-formula obtained from the expansion. The result is a tableau with two open branches:

1.	$T\,\text{even}(0)$	*assumption*
2.	$T\,\forall x(\text{odd}(x) \rightarrow \text{even}(s(x)))$	*assumption*
3.	$F\,\text{odd}(s(s(s(0))))$	*assumption*
4.	$T\,\text{odd}(s(0)) \rightarrow \text{even}(s(s(0)))$	γ on 2
5.	$T\,\text{odd}(s(0))$	
6.	$T\,\text{even}(s(s(0)))$	$(T \rightarrow)$ on 4, 5
7.	$F\,\text{odd}(s(0))$	

The open left branch, from lines 6 and 3 gives us the hypothesis $H_1 = \text{even}(s(s(0))) \rightarrow \text{odd}(s(s(s(0))))$ and the open right branch, from lines 1 and 1 gives us $H_2 = \text{even}(0) \rightarrow \text{odd}(s(0))$, so that the abduced hypothesis is $H = H_1 \wedge H_2$, which is a form of low compromise and correct abduced formula.

Clearly, any formula that implies H is also a possible solution. One such possibility is to apply reverse skolemization on H, but not on its parameters, for which there are none, but on its constants, namely, on the constant 0. This is clearly a correct possibility, for parameters behave as constants. In this case, the left open branch would yield $H'_1 = \forall x(\text{even}(s(s(x))) \rightarrow \text{odd}(s(s(s(x)))))$ and the right open branch would yield $H'_2 = \forall x(\text{even}(x) \rightarrow \text{odd}(s(x)))$. Clearly, H'_2 entails (or subsumes) H'_1, so in this case of reverse skolemization of constants, we would obtain the abduced hypothesis $H' = H'_2$, as desired.

In this example, the crucial step is the γ expansion on line 4 with $x = s(0)$, which admittedly was 'good guess'. So the example shows that our method *can* abduce the desired hypothesis, but no computational rule is provided to guarantee that it *will* infer the desired hypotesis.

5.1 Admissibility

In Algorithm 5.1, line 4 states that we have to choose an *admissible* formulas $\Phi_i \subseteq \Psi_i$ on which the abduced hypothesis H is built. For the purposes of this article[3], *admissibility* is defined as follows:

- $H \not\vdash G$: the hypothesis does not prove the data independently of the background data; and

[3]There is no consensus on what constitutes a 'good' explanation.

- $\Gamma, H \vdash \bot$ only if $\Gamma \vdash \bot$: the hypothesis contradicts the background data only if the background data is itself contradictory.

In traditional abduction the second condition is simply $\Gamma, H \nvdash \bot$, as the usual condition that $\Gamma \nvdash G$ implies that $\Gamma \nvdash \bot$.

The problem with admissibility testing in first-order abduction is, as usual, the undecidability of the test. To circumvent this problem, we propose certain restrictions on $\Phi_i \subseteq \Psi_i$, namely:

- Φ_i must not contain only subformulas of G; this aims at avoiding $H \vdash G$;
- Φ_i must not contain only the conjugate of subformulas of Γ; this aims at avoiding $\Gamma, H \vdash \bot$.

Note that these are simply heuristic rules to achieve real admissibility. When there are several subsets of Ψ to be chosen, such heuristics shrinks the search space. However, a problem occurs when such conditions cannot be met. In that case, a possible solution is to realise that the tableau must be further expanded before Algorithm 5.1 can be applied.

6 Dynamic abduction and proof construction

So far, all the examples presented have dealt with traditional abduction. That is, the sequents presented in the examples were simple enough for us to know that they were not provable and the abduced hypothesis computed using Algorithm 5.1 did in fact close the tableau.

That situation is in fact the exceptional one in terms of first-order proving. In general, one does not know a priori if the tableau closes or not. As the abduction method in Algorithm 5.1 does not specify when to stop expanding a tableau for the original sequent $\Gamma \vdash^? \Delta$, one can arbitrarily interrupt tableau expansion and apply Algorithm 5.1, computing a formula H' such that, by Theorem 1, it is guaranteed that

$$\Gamma, H' \vdash_{\mathrm{FOL}} \Delta.$$

That is, there is a closed tableau for this sequent. However, we have not established if the original sequent is in fact provable or not.

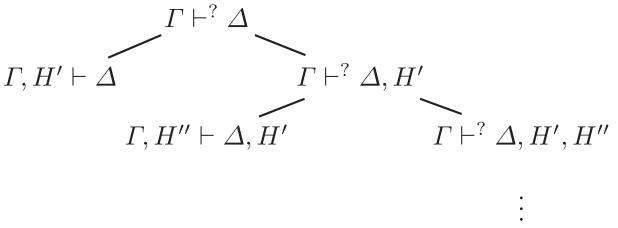
The idea of *dynamic abduction* is to use the information obtained so far to compose a proof for the original sequent, or eventually to generate a countermodel for it. The composition of proofs is done via a cut rule:

$$\dfrac{\Gamma \vdash^? \Delta}{\Gamma, H' \vdash \Delta \qquad \Gamma \vdash^? \Delta, H'}$$

Note that in general this cut rule may be *non-analytic*, that is, H' may not be a subformula of either Γ or Δ. The left sequent is a provable one, but the provability of the right sequent remains undecided, $\Gamma \vdash^? \Delta, H'$. If this latter sequent is in fact provable, so is the original sequent. Conversely, if there is a countermodel for this latter sequent, it is also a countermodel for the original one; indeed, such a countermodel should satisfy Γ and falsify both Δ and H', thus asserting also that $\Gamma \nvdash \Delta$.

This means that the decision of $\Gamma \vdash^? \Delta$ is completely determined by the decision of $\Gamma \vdash^? \Delta, H'$. As this latter sequent has one extra hypothesis in respect to the original one, it is expected that a proof for it is no larger than a proof for the original sequent. We can now proceed with a partially expanded tableau for $\Gamma \vdash^? \Delta, H'$, and interrupt it at some point to apply again Algorithm 5.1. If we

recursively compose the results, we obtain



As before, the decision of $\Gamma \vdash^? \Delta, H', H''$ leads to the the decision of its direct ancestor sequent, which recursively leads to the decision of the original sequent. This constitutes the basis for a recursive procedure that characterises the dynamic abduction process illustrated in Algorithm 6.1.

Algorithm 6.1 *DynamicFOLAbduction*(Γ, Δ)

Input: A sequent $\Gamma \vdash^? \Delta$
Output: A proof or countermodel for it; or simply *no decision*

1: $\Delta_1 = \Delta$
2: **for** $i = 1$ to LIMIT **do**
3: Let $\mathcal{T}_i$ be a partially expanded tableau for $\Gamma \vdash^? \Delta_i$
4: **if** $\mathcal{T}_i$ is closed **then**
5: **return** a non-analytic proof $\Gamma \vdash \Delta$.
6: **else if** $\mathcal{T}_i$ has a saturated open branch **then**
7: **return** a countermodel for $\Gamma \nvdash \Delta$.
8: **else**
9: $H_i = FOLAbduction(\Gamma, \Delta_i, \mathcal{T}_i)$
10: $\Delta_{i+1} = \Delta_i \cup \{H_i\}$
11: **end if**
12: **end for**
13: **return** *no decision*

It is important to note that Algorithm 6.1 is *not* guaranteed to terminate, unlike its propositional counterpart presented in [12]. In fact, if it always terminated we would have a decision procedure for first-order logic, which is impossible. Therefore, we impose a limit number of iterations of iterations which is a parameter to the algorithm, LIMIT. When such limit number of iterations is reached, the procedure terminates without a decision to output. Note that, at this point, the algorithm could actually output a sequent $\Gamma \vdash^? \Delta_i$ that is decidable iff the input sequent is.

Note again that there is an unspecified step in line 3 of Algorithm 6.1, namely how to interrupt the expansion of the tableau. In [12], several strategies were discussed with implications on the size of the computed proof, when one does exist. The proof for the original sequent, when a closed tableau is reached, is composed via several cut rules, as explained above, thus consisting in a non-analytic first-order proof. As the underlying proof method is a KE-tableau, it can represent those non-analytic cuts by means of applications of the PB rule, so the proof generated can be presented as a simple KE-tableau proof. On the other hand, if some of the generated branches are open and saturated, a countermodel for the current $\Gamma \vdash \Delta_i$ sequent can be computed, which is also a countermodel for the input sequent as explained above.

In the light of those properties, we define the *weak correctness* of an algorithm as the property that, if the algorithm stops, its output is a correct one. We have thus the following result.

THEOREM 2 (Weak correctness of Algorithm 6.1)
If Algorithm 6.1 stops, its output is a correct one.

EXAMPLE 7
Consider the (valid) sequent $\vdash \exists x \forall y m(x,y) \rightarrow \forall y \exists x m(x,y)$. Consider a tableau for it in which only δ-expansions are applied.

$$\begin{array}{l} T\,\exists x \forall y m(x,y) \\ F\,\forall y \exists x m(x,y) \\ \hline T\,\forall y m(a,y) \\ F\,\exists x m(x,b) \end{array}$$

and suppose that the tableau is interrupted at this point. According to Algorithm 5.1, the abduced hypothesis is $H = \forall u \forall w(\forall y m(u,y) \rightarrow \exists x m(x,w))$. Adding H as an extra hypothesis clearly closes the tableau, but it is not a very informative formula for future steps in a proof construction. In fact, the formula H is logically equivalent to the original formula, and its use in the dynamic proof construction would be pointless.

This example illustrates that dynamic abduction is not a panacea, and one has to consider serious strategies in order to start applying it in proof search. A promising idea towards obtaining such a strategy comes from the work of Nabeshima *et al*. [20] with the application of *production fields* to limit the hypothesis that can be generated; production fields are a generic form of constraint over the *clauses* that can be abduced by SOLAR, with the capacity of limiting both the format of its literal as well as other clausal properties, such as its length. The study of effective strategies for dynamic abduction is a topic for further work; a discussion on the topic for propositional logic can be found in [12].

7 Conclusions and further work

We have shown a generalized form of abduction that is cut-based and that is capable of performing automated first-order abduction without having to rely on undecidable preconditions of traditional abduction. The correctness of the method was demonstrated. When the input sequent is invalid, the method performs in a similar way to other tableaux-based abductive methods in the literature, and oftentimes more efficiently, relying on smaller KE-style proofs. When applied to potentially provable sequents, the method can be dynamically combined to generate proofs for a provable input sequent.

A remarkable property of the method above is that it presents a way of computing non-analytic proofs for first-order theorems. And the whole method of dynamic abduction would not have been possible had we insisted on analytic proofs only.

The method proposed here is not the only non-analytic inference system to be used. For sure, a good deal of the literature on automated abduction was developed in terms of Resolution, which clearly is a non-analytic method [2]. However, unlike resolution, the non-analytic method proposed here is actually the basis for the composition of several abductive steps, a feature not found in resolution-based methods.

7.1 Suggestions for further work

Resolution-based methods use in a fundamental way the easiness for dealing with quantified variables provided by the unification procedure. The combination of unification with tableau inference systems have been best realized in the literature with the *disconnection tableaux* [3, 17]. However, such method is still based on clausal-form first-order logic and is still very much based on analytic methods.

An amalgamation of the techniques in the disconnection tableaux and those of dynamic abduction described in this section remains for future work.

Also, one can expand the first-order rules to deal with equality as a logical connective. There are special rules to do that in analytic tableaux and so also in KE-tableaux. However, as there are special ways of dealing with equality in disconnection tableaux, it may be particularly interesting for future work to combine the cut-based abductive method with the treatment of equality in disconnection tableaux.

Funding

Partly supported by CNPq grant PQ 301294/2004-6 and FAPESP project 2008/03995-5.

References

[1] A. Aliseda-Llera. Seeking explanations: abduction in logic, philosophy of science and artificial intelligence. PhD Thesis, Stanford University, Stanford, CA, USA, 1997.

[2] L. Bachmair and H. Ganzinger. Resolution theorem proving. In *Handbook of Automated Reasoning*. A. Robinson and A. Voronkov, eds, Vol. I. pp. 19–99. Elsevier Science, 2001.

[3] J. P. Billon. The disconnection method — a confluent integration of unification in the analytic framework. In *TABLEAUX*. P. Miglioli, U. Moscato, D. Mundici, and M. Ornaghi, eds, Vol. 1071 of *Lecture Notes in Computer Science*, pp. 110–126. Springer, 1996.

[4] G. Boolos. Don't eliminate cut. *Journal of Philosophical Logic*, **13**, 373–378, 1984.

[5] S. R. Buss. Some remarks on lengths of propositional proofs. *Archive for Mathematic Logic*, **34**, 377–394, 1995.

[6] A. Carbone and S. Semmes. *A Graphic Apology for Symmetry and Implicitness. Oxford Mathematical Monographs*. Oxford University Press, 2000.

[7] W. Carnielli. Surviving abduction. *Logic Journal of IGPL*, **14**, 237–256, 2006.

[8] S. Colucci, T. Di Noia, E. Di Sciascio, F. M. Donini, and M. Mongiello. A uniform tableaux-based approach to concept abduction and contraction in ALN. In *Description Logics*, V. Haarslev and R. Möller, eds, vol. 104 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2004.

[9] M. D'Agostino. Are tableaux an improvement on truth-tables? — Cut-free proofs and bivalence. *Journal of Logic, Language and Information*, **1**, 235–252, 1992.

[10] M. D'Agostino. Tableau methods for classical propositional logic. In *Handbook of Tableau Methods*. M. D'Agostino, D. Gabbay, R. Haehnle, and J. Posegga, eds, pp. 45–124. Kluwer, 1999.

[11] M. D'Agostino and M. Mondadori. The taming of the cut. Classical refutations with analytic cut. *Journal of Logic and Computation*, **4**, 285–319, 1994.

[12] M. D'Agostino, M. Finger, and D. Gabbay. Cut-based abduction. *Logic Journal of the IGPL*, **16**, 537–560, 2008.

[13] J. de Kleer. An assumption-based tms. *Artificial Inteligence*, **28**, 127–162, 1986.
[14] K. Eshghi and R. A. Kowalski. Abduction compared with negation by failure. In *ICLP*, pp. 234–254, 1989.
[15] D. Gabbay and J. Woods. Advice on abductive logic. *Logic Journal of the IGPL*, **14**, 189–219, 2006.
[16] A. C. Kakas, R. A. Kowalski, and F. Toni. Abductive logic programming. *Journal of Logic and Computation*, **2**, 719–770, 1992.
[17] R. Letz and G. Stenz. The disconnection tableau calculus. *Journal of Automated Reasoning*, **38**, 79–126, 2007.
[18] M. C. Mayer and F. Pirri. First-order abduction via tableau and sequent calculi. *Journal of the IGPL*, **1**, 99–117, 1993.
[19] M. C. Mayer and F. Pirri. Modal propositional abduction. *Journal of the IGPL*, **3**, 99–117, 1995.
[20] H. Nabeshima, K. Iwanuma, K. Inoue, and O. Ray. Solar: an automated deduction system for consequence finding. *AI Communications*, **23**, 183–203, 2010.
[21] D. Poole. Representing knowledge for logic-based diagnosis. In *Proceedings of the International Conference on Fifth Generation Computer Systems*, pp. 1282–1290, 1988.
[22] D. Poole. A logical framework for default reasoning. *Artificial Intelligence*, **36**, 27–47, 1988.
[23] R. Reiter and J. de Kleer. Foundations of assumption-based truth maintenance systems: Preliminary report. In *AAAI*. pp. 183–189, 1987.
[24] A. L. Reyes-Cabello, A. Aliseda-Llera, and A. Nepomuceno-Fernández. Towards abductive reasoning in first-order logic. *Logic Journal of the IGPL*, **14**, 287–304, 2006.
[25] M. Shanahan. Prediction is deduction but explanation is abduction. In *Proceedings of IJCAI*. pp. 1055–1060, 1989.
[26] R. M. Smullyan. *First-Order Logic*. Springer, 1968.
[27] A. Yamamoto. Which hypotheses can be found with inverse entailment? In *7th International Workshop on Inductive Logic Programming (ILP97)*. pp. 296–308. Springer, 1997.

Received 18 November 2010