Available online at www.sciencedirect.com
ScienceDirect
Electronic Notes in Theoretical Computer Science
ELSEVIER
Electronic Notes in Theoretical Computer Science 247 (2009) 39–49
www.elsevier.com/locate/entcs

Using Abduction to Compute Efficient Proofs

Marcelo Finger[1,2]

Department of Computer Science
University of Sao Paulo
Sao Paulo, Brazil

Abstract

The aim of this work is to show how to compute an extra hypothesis H to an unproved sequent $\Gamma \vdash^{?} G$, such that:

- $\Gamma, H \vdash G$ is provable;
- H is not trivial.
- If $\Gamma \vdash G$ (which is not known a priory) then $\Gamma, H \vdash G$ has a much simpler proof.

Due to the last item, this is *not exactly* the usual abductive reasoning found in literature, for the latter requires the input sequent not to be derivable.
The idea is that Γ is a contextual database, containing background knowledge, G is a goal formula representing some fact or evidence that one wants to explain or prove, and H is an hypothesis that explains the evidence in the presence of the background knowledge or that facilitates the proof of G from Γ.
We show how this task is related to the problem of computing non-analytic cuts. Several algorithms are provided that compute efficient proofs with non-analytic cuts via abductive reasoning.
This is a joint work with Marcello D'Agostino and Dov Gabbay.

Keywords: Abductive reasoning, proof efficiency, non-analytic cuts, sequent calculus.

1 Introduction

Abductive reasoning, as usually found in the literature, is concerned with computing an *explanation*, or a extra *hypothesis*, such that, given a set of background data and a goal, the data together with the extra hypothesis prove the goal. In formal terms, given the data Γ and the goal G such that $\Gamma \nvdash G$, the abduction process produces a formula H such that $\Gamma, H \nvdash G$[9,8,10,5,2,1,6].

The literature provides certain restrictions to a hypothesis H to be acceptable as an explanation. In our case, we fix the following constraints.

[1] Partly supported by CNPq grant PQ 304607/2007-0 and FAPESP project 04/14107-2.
[2] Email: mfinger@ime.usp.br

doi:10.1016/j.entcs.2009.07.047

- $\Gamma, H \vdash G$ is provable;
- if $\Gamma \cup \{G\}$ is consistent then $\Gamma \cup \{H\}$ is consistent; and $H \not\vdash G$, that is, H alone does not imply G.

Although no underlying logic is fixed for the notion of abduction to be defined, in this paper we consider only classical propositional logic.

Our task, however, is *not exactly* the usual abductive reasoning found in the literature. It is in fact a generalisation of the traditional abduction task, simultaneously covering two cases:

(a) if $\Gamma \not\vdash G$, the problem reduces to traditional abduction, which we call *hypothesis generation*. In generating H, we will search for a compromise between minimality and computational efficiency, which is, in fact, in accordance with Peirce's perception of "best explanation", as he introduced the notion of abductive reasoning [7].

(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 $\Gamma \vdash G$, where "simpler" may mean "shorter" or just "easier to grasp". There is also a compromise to be reached between finding a simpler proof and finding a proof at a small computational cost.

We are expanding the traditional *explanationist* view of abduction, with a*simplificationist* view of abduction, that is, a capacity of providing a simpler or more compact account of facts. This extended view of abduction makes sense if the validity of $\Gamma \vdash G$ in not known in advance, as in traditional theorem proving.

This process of computing extra hypothesis to a provable sequent is closely related to computing non-analytic cuts. The composition of several abduction steps leads to what we call *dynamic abduction*, in which a proof or refutation of a given sequent $\Gamma \vdash^{?} G$ is obtained by composing the abductive steps with non-analytic cuts in sequent proofs.

This paper presents the algortihmic side of generalized abduction applied to computing proofs, with a discussion for associated heuristics. A larger version with a broader discussion of the method is in preparation [3].

We present a technique for computing an extra hypothesis based on tableau method. The same technique is used for both cases: $\Gamma \vdash G$ and $\Gamma \not\vdash G$. As Smullyan's Analytic Tableaux [11] is based on a cut-free version of the sequent calculus, we employ the KE tableau method which is able to efficiently simulate sequent calculus with full use of cuts [4]. This method can be dynamically iterated, so as to compute non-analytic proofs. Those proofs tend to be shorter than analytic one. An example is presented showing how polynomially sized proofs can be computed according to this method for a subclass of Tseitin formulas.

The paper is structured as follows. Initially, the concepts related to KE-tableau in Section 2. The basic KE-tableau abductive procedure is presented in Sectionsec:branch. It is then shown how this method can be iterated to compute proofs

with non-analytic cuts in Section 4. An example of how the method can be used to compute polynomially sized proofs of a (limited) class of "hard" tautologies, namely a subclass of Tseitin Formulas, is shown in Section 5. Section 6 then concludes the paper.

2 Preliminaries

We consider formulas built over a countable set of propositional atoms denoted by the lower case letters p, q, r, etc., and connectives $\neg$, $\wedge$, $\vee$ and $\rightarrow$. We represent formulas by upper case Latin letters: A, B, C, etc. We represent sets of formulas by upper case Greek letters, such as Γ, Δ, Φ and Ψ. We write Γ, A to represent $\Gamma \cup \{A\}$ and Γ, Δ to represent $\Gamma \cup \Delta$.

A *sequent* is an expression of the form $\Gamma \vdash \Delta$, where Γ is the antecedent and Δ is the succedent. A sequent inference rule allows one to infer a sequent $\mathcal{S}$ from zero or more sequents $\mathcal{S}_1, \ldots, \mathcal{S}_k$. If the inference rule has 0 premisses, it is called a sequent *axiom*. A k-premissed inference rule, with $k > 0$, is called *analytic* if every formula in the k premisses occurs as a subformula in the conclusion $\mathcal{S}$. In most sequent calculi, all rules are analytic, except for the *cut rule*, which we assume to have the following format

$$\frac{\Gamma_1 \vdash \Delta_1, A \qquad A, \Gamma_2 \vdash \Delta_2}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2}\ (\text{Cut})$$

The formula A is called the *cut formula* of this inference and the cut is analytic if A occurs in $\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2$. A sequent proof is a tree whose nodes are sequents, having sequent axioms at its leaves and such that every internal node is obtained by the application of some k-premissed inference rule, $k > 0$. A proof is analytic if it only uses analytic inferences. We assume that the only potentially non-analytic rule is cut, so that a cut-free proof is always analytic. We write $\Gamma \vdash^{?} \Delta$ when we do not know if $\Gamma \vdash \Delta$ is provable.

KE-tableaux were proposed by D'Agostino and Mondadori [4] incorporating the principle of excluded middle or, at the semantic level, to the *principle of bivalence* (PB). This principle is the tableau correspondent of the cut rule in the sequent calculus.

KE-tableaux deal with *signed* formulas. If A is a formula, $T\ A$ and $F\ A$ are signed formulas. $T\ A$ is the *conjugate formula* of $F\ A$, and vice versa; if $X \in \{T, F\}$ then $\bar{X}$ is defined as follows: $\bar{T} = F$ and $\bar{F} = T$. Each connective is associated with a set of *linear expansion rules* also called *elimination rules*. 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*. Figure 1 shows the KE linear linear expansion rules for classical logic. The *only* branching rule in KE is the *Principle of Bivalence* (PB), stating that a formula A must be either true or false, as illustrated in Figure 2.

The expansion of KE-tableau for the sequent $A_1, \ldots, A_n \vdash B_1, \ldots, B_m$ starts

$$\frac{T\ A \wedge B}{\begin{array}{l} T\ A \\ T\ B \end{array}}(T\wedge) \qquad \frac{F\ A \wedge B}{T\ A}(F\wedge_1) \qquad \frac{F\ A \wedge B}{T\ B}(F\wedge_2)$$

$$\frac{T\ A \vee B}{F\ A}(T\vee_1) \qquad \frac{T\ A \vee B}{F\ B}(T\vee_2) \qquad \frac{F\ A \vee B}{\begin{array}{l} F\ A \\ F\ B \end{array}}(F\vee)$$

$$\frac{T\ A \to B}{\begin{array}{l} T\ A \\ T\ B \end{array}}(T\to_1) \qquad \frac{T\ A \to B}{F\ B}(T\to_2) \qquad \frac{F\ A \to B}{\begin{array}{l} T\ A \\ F\ B \end{array}}(F\to)$$

$$\frac{T\ \neg A}{F\ A}(T\neg) \qquad \frac{F\ \neg A}{T\ A}(F\neg)$$

Fig. 1. KE Expansion Rules

Fig. 2. Principle of Bivalence

with a single branch containing $TA_1, \ldots, TA_n, FB_1, \ldots, FB_m$. A branch expansion step is allowed when the premisses of an expansion rule are present in the branch and results in adding the conclusions of the rule to the end of that branch. The PB branching rule splits a branch into two.

The goal of the expansion is to obtain a *complete* KE-tableau. The tableau is *complete* if every branch in it is complete. A branch is *complete* if it is either *closed* or *saturated*.

A branch is *closed* if it contains $F\ A$ and $T\ A$ for some A; otherwise it is *open*. The tableau is *closed* if all its branches are closed; a closed tableau is complete.

A branch is *saturated* if it is open and all non-atomic signed formulas in it are *fulfilled*. A signed formula is *fulfilled* in a branch if it is the main formula of an instance of an expansion rule such that the conclusions of that instance already occur in the branch. Clearly, if a formula is the main formula of an expansion it becomes fulfilled; however, it is possible for a non-atomic signed formula to become fulfilled by the expansion of other signed formulas as well. Note that any valuation that satisfies the conclusions of an instance of an expansion rule also satisfies its main formula, so that fulfilled formulas are "subsumed" by other formulas in the branch. By definition, a signed atomic formula is always unfulfilled.

The deductibility relation $\vdash_{\mathbf{KE}}$ is defined as follows:

$$A_1, \ldots, A_n \vdash_{\mathbf{KE}} B_1, \ldots, B_m \text{ iff there is a closed KE-tableau for } TA_1, \ldots, TA_n, FB_1, \ldots, FB_m. \quad (1)$$

Any saturated branch provides a counter-valuation for the sequent.

3 A KE-tableau Abductive Procedure

The basic step of the method is to abduce a signed formula based on an open branch, which may or may not be saturated. The branch is viewed as a set of signed formulas, and a formula is abduced so as to close the branch.

Given a nonempty set of signed formulas $\Phi = \{TA_1, \ldots, TA_n, FB_1, \ldots, FB_m\}$, we compute the hypothesis $H(\Phi)$ as follows:

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

Lemma 3.1 *Given a KE-tableau branch containing formulas Φ, if we add $TH(\Phi)$ as a top hypothesis, then this branch can be expanded into a closed subtree.*

If the initial tableau has more than one open branch, then we have to apply this method to each branch. The final abduced formula is the conjunction of the formulas computed for each branch. This idea is presented in Algorithm 1, which presents the branch-driven abduction algorithm. Note that it is a non-deterministic algorithm in a twofold way. First, it lets one expand the tableau in whatever fashion one wants, and the halting of this expansion is not specified; this non-determinism is in fact hidden in the fact that a partially expanded tableau is given *as input* to the abduction process. Second, one can choose the subset Φ of unfulfilled signed formulas to generate the abduced hypothesis.

Theorem 3.2 (Correctness of Algorithm 1) *Algorithm 1 is correct, that is, on input $\Gamma \vdash^{?} G$ it outputs a formula H such that $\Gamma, H \vdash G$.*

4 Cut-Based Dynamic Abduction

Before we describe the dynamic abduction procedure, consider the following result, which is used extensively by it. This result is based on the application of the Cut Rule to a pair of sequent; similarly, it can be obtained by the application of PB to combine a pair of KE-tableaux.

Algorithm 1 Branch-driven Abduction

BranchAbduction$(\Gamma, G, \mathcal{T})$

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 $\mathcal{T}$.
2: **for** $i = 1$ to k **do**
3: Let $\Psi_i = \{XA \in \mathcal{B}_i | XA$ unfulfilled in $\mathcal{B}_i\}$
4: Choose $\Phi_i \subseteq \Psi_i$
5: Let $H_i = H(\Phi_i)$
6: **end for**
7: **return** $H_1 \wedge \ldots \wedge H_k$

Lemma 4.1 *Suppose* $\Gamma, H \vdash G$. *Then* $\Gamma \vdash G$ *iff* $\Gamma \vdash H, G$.

The dynamic process of repeated applications of the branch abduction algorithm is illustrated in Figure 3.

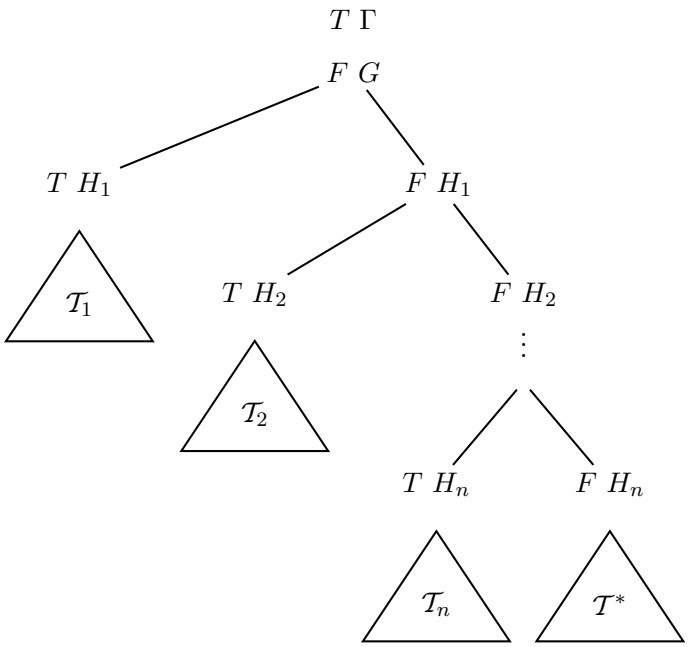


Fig. 3. The Dynamic Abduction Process

The idea of the construction on Figure 3 is the following. One starts with the construction of a proof for the original sequent $\Gamma \vdash^{?} G$. Suppose there is some method for deciding how to start building a finite tableau $\mathcal{T}_1$ for it; it can be either atom elimination, or approximated reasoning, or any decidable method. If $\mathcal{T}_1$, then the proof is finished. Otherwise, the abduction Algorithm 1 is applied, yielding an abduced formula H_1. The correctness of the algorithm guarantees that an extension

of tableau $\mathcal{T}_1$ for $\Gamma, H_1 \vdash G$ closes. Furthermore, Lemma 4.1 guarantees that the original sequent is provable iff $\Gamma \vdash H_1, G$ is, so the proof proceeds by constructing a tableau for $\Gamma \vdash^? H_1, G$; the KE-method, guarantees there is a proof or refutation for it no longer than a proof or refutation for the original sequent. A proof for the original sequent can be composed with a single application of a potentially non-analytic cut, ie a branching over the abduced formula H_1.

This process can then be iterated, as illustrated in Figure 3. Tableaux $\mathcal{T}_1, \ldots, \mathcal{T}_n$ all close due to the correctness of Algorithm 1. After every abduction step i, only the rightmost branch, namely the one containing FH_i, is open. At the end of the proof, the rightmost tableau $\mathcal{T}^*$ closes iff the initial sequent is provable, due to iterated applications of Lemma 4.1. When this process terminates we have computed potentially non-analytic cut formulas $H_1, \ldots, H_n$, generating a non-analytic proof or refutation for the input sequent.

The dynamic abduction algorithm is shown in Algorithm 2. The idea is to parameterise it with an abduction heuristic $\mathcal{H}$. Note that the final product is a non-analytic proof for the original sequent.

Algorithm 2 Non-analytic Tableau Proof via Dynamic Abduction

DynamicAbduction($\Gamma, G, \mathcal{H}$)

Input: A sequent $\Gamma \vdash^? G$ and abduction heuristics $\mathcal{H}$.
Output: A tableau $\mathcal{T}$ for $\Gamma \vdash G$ or a counter-valuation

1: $i := 1$
2: *openBranch* $:= \{TA | A \in \Gamma\} \cup \{FG\}$
3: $\mathcal{T} :=$ *openBranch*
4: **while** *true* **do**
5: $\quad \mathcal{T}_i :=$ apply abduction heuristics $\mathcal{H}$ to *openBranch*
6: $\quad$ **if** $\mathcal{T}_i$ closes **then**
7: $\quad\quad$ Attach $\mathcal{T}_i$ to the end of the open branch in $\mathcal{T}$
8: $\quad\quad$ **return** $\mathcal{T}$
9: $\quad$ **else if** there is a saturated open branch in $\mathcal{T}_i$ **then**
10: $\quad\quad$ **return** a counter valuation obtained from the open branch
11: $\quad$ **else**
12: $\quad\quad$ $H_i :=$ BranchAbduction(*openBranch*, $\mathcal{T}_i$)
13: $\quad\quad$ $\mathcal{T}_i :=$ expand and close $\mathcal{T}_i \cup \{TH_i\}$
14: $\quad\quad$ Expand $\mathcal{T}$ with an application of PB. On the left add TH_i and $\mathcal{T}_i$. On the right add FH_i
15: $\quad\quad$ *openBranch* $:=$ *openBranch* $\cup \{FH_i\}$
16: $\quad$ **end if**
17: $\quad i := i + 1$
18: **end while**

At each iteration step i, a tableaux $\mathcal{T}_i$ is expanded as an application of the abduction heuristics to $\Gamma \vdash^? H_1, \ldots, H_{i-1}, G$ *without using the abduced hypothesis* H_i. In fact, H_i can only be computed *after* $\mathcal{T}_i$ is expanded. When $\mathcal{T}_i$ is expanded, there are three possibilities:

- The analytic tableau for $\mathcal{T}_i$ closes (line 6). Then a proof has been constructed for the original sequent $\Gamma \vdash G$.
- The analytic tableau for $\mathcal{T}_i$ has an open saturated branch (line 9). Then a counterexample for $\Gamma \nvdash G$ has been obtained.
- $\mathcal{T}_i$ is open, with no saturated branch. In this case, we can apply Algorithm 1 and compute H_i (line 12). By the correctness of the procedure, $\mathcal{T}_i$ can be closed adding H_i (line 13), so $\mathcal{T}_i$ is expanded an analytic closed tableau for $\Gamma, H_i \vdash H_1, \ldots, H_{i-1}, G$.

The two initial cases are the termination cases of the non-analytic proof. In the last case, the construction of the proof can continue, such that $\mathcal{T}_i$ is a closed sub-tableau. In general, the abduced formula H_i is not a subformula of the original sequent, so the process is very likely to generate a non-analytic proof.

Theorem 4.2 (Partial correctness of Algorithm 2) *If Algorithm 2 stops, then either it produces a proof of* $\Gamma \vdash G$ *or it produces a counter-valuation for it.*

Termination depends on the abduction heuristics chosen. If the abduction heuristics has the potential of generating infinite proofs, this process can always be interrupted. In this case, a fixed number of iterations k may be also given as part of the heuristics, such that the abduction procedure can be replaced by a simple analytic KE tableau expansion for $\Gamma \vdash^{?} H_1, \ldots, H_k, G$. This last expansion generates the tableau $\mathcal{T}^*$ is Figure 3 and it is guaranteed to always terminate.

4.1 Abduction Heuristics

An *abduction strategy* or *abduction heuristics* is a procedure that allows one to choose the abduction parameters. Such abduction heuristics has to decide:

- When to apply abduction?
- Which formula do abduce?

A naive heuristics can be described by:

- When to apply abduction?
  After applying all linear expansion rules
- Which formula do abduce?
  The least compromising formula. That is, for each branch compute $H(\Phi)$, where Φ is the set of all unfulfilled formulas in the branch.

However, this form of heuristics is non-terminating, for the tableau has the same unfulfilled formulas after applying abduction as it had before.

This means that abduction is not a panacea. It can give extra hypotheses that make the original proof more efficient, but a naive approach will not guarantee a more efficient proof for the input sequent, $\Gamma \vdash G$.

A better abduction heuristics is given as follows. Instead of building a normal tableau, several subsets of the signed formulas will be chosen after an initial linear

saturation of the tableau. Each subset will not generate in general a closed tableau; if one subset generates a closed tableau, the search space shrinks. So the abduction procedure is applied to the complete tableau generated by a subset of the original formulas, such that:

- The formulas in a selected subset must have some atoms in common.
- The abduction process will *not* compute the least compromising formula. The aim is to eliminate from the abduced formulas some or all of the common atoms, so as to promote a reduction in the "dimension" of the problem. This can be done safely whenever such atom occurs only in the selected subset of formulas, and nowhere else in that branch.

We call it the *subformula elimination heuristics*, which can be described by:

- When to apply abduction?
  Choose a set of subformulas $\{A_1, \ldots, A_k\}$ to be eliminated, construct a complete tableau for all formulas containing some $A_i, 1 \leq i \leq k$.
- Which formula do abduce?
  For each open branch $\mathcal{B}$, compute $H(\Phi)$, where Φ is the set of unfulfilled formulas in $\mathcal{B}$ not containing some $A_i, 1 \leq i \leq k$.

Clearly, when the eliminated subformulas are atoms, the elimination heuristics is terminating, for there are only finitely many atoms to eliminate.

Note that a generic sequence of elimination steps, each of which eliminates a set of atoms or subformulas from the sequent, has the potential of producing a multiplicative increase in the size of the resulting abduced formula, which can lead to an exponential explosion on the size of the proof.

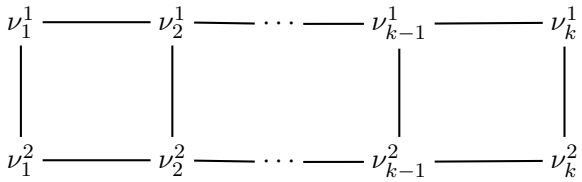
However, for a very well know class of "hard" formulas, this heuristics is capable of producing short proofs.

5 An Example

We consider here Tseitin Formulas, for which it is know that there does not exit proof of polynomial size with respect to the number of atoms, if only clausal resolution is applied [12].

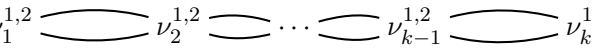
In fact, we are going to consider here a *Tseitin sequent* $\Gamma \vdash \Delta$ constructed as follows. Consider any undirected graph $G = \langle N, E \rangle$, where N is the set of nodes, $|N| = n$, and the E is the set of edges, $|E| = m \leq n(n-1)$; it is possible that more than one edge connects two nodes. Every edge is associated to a distinct atom p_j, $1 \leq j \leq m$, and every node is associated to a formula A_k, $1 \leq k \leq n$, namely the exclusive-or of all adjacent edges. An odd number of nodes receive marks; usually, a single marked nodes suffice. Let Γ be the set of formulas associated to marked nodes, and let Δ be the set of formulas associated to unmarked nodes. The sequent $\Gamma \vdash \Delta$ is always provable, a fact associated to the property that the sum of degrees of all nodes in a graph is even. In fact, each propositional valuation represents a subgraph of G, to which the property also applies.

We consider here a subclass of Tseitin Graphs, namely $2, k$-grids, which are graphs of the following format:



Theorem 5.1 *Let $\Gamma \vdash \Delta$ be a Tseitin sequent based on a $2, k$-grid graph $G = \langle N, E \rangle$ with an odd number of marked nodes. Then the the subformula elimination heuristics produces proof of polynomial size on the size of $\Gamma \vdash \Delta$.*

The proof is sketched as follows. Let $G_0 = G$. There are two phases. In the first phase, each abduction step i transforms G_{i-1} into G_i with one fewer node a one or more fewer edges. This is done by choosing a pair of adjacent nodes, ν_i^1 and ν_i^2, and fusing them into $\nu_i^{1,2}$, eliminating the edges with extremities on ν_i^1 and ν_i^2. After k steps, the result is the following graph.



In this process, no node has degree above 4, so the formulas computed by abduction with subformula elimination heuristics, which associates a formula for each node $\nu_i^{1,2}$, $1 \leq i \leq k$, have at most 2^4 atoms, so are of bounded size with a fixed bound; to abduce each such formula, there are two formulas in the initial tableau with two branches each of depth at most m. As there are k node fusion in this phase, this part of the proof is $O(km)$ in space.

The second phase consists of fusing two adjacent nodes in the graph at the end of phase 1, shown above. This corresponds to performing abduction to eliminate the two proposition symbols representing the two edges between the nodes. When a single node is obtained, the tableau is closed. The formulas computed by abduction have also at most 2^4 atom occurrences. Each abduction step produces a tableau with at most four branches of depth at most m. So the second phase of the proof is also $O(km)$ in size. So the constructed tableau is polynomial in size with the number of atoms of the initial problem, times k, which finishes the proof.

A method that deals with any kind of Tseitin formula is currently under analysis.

6 Conclusion

A generalized abduction method was presented which allows for the computation of non-analytic cuts in KE-tableau proofs. The method presented was capable of generating polynomially sized proofs to some limited class of "hard" theorems. Future work intends to explore the application of these methods, with possibly new abduction heuristics, to a larger class of problems.

References

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

[2] Marta Cialdea Mayer and Fiora Pirri. Abduction is not deduction-in-reverse. *Journal of the IGPL*, 41(1):95–108, 1996.

[3] Marcello D'Agostino, Marcelo Finger, and Dov Gabbay. Cut-based abduction. *Logic Journal of the IGPL*. Accepted, doi:10.1093/jigpal/jzn020.

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

[5] Robert Demolombe and Luis Fariñas del Cerro. An inference rule for hypothesis generation. In *Proceedings of IJCAI*, pages 152–157, 1991.

[6] Dov Gabbay and John Woods. Advice on abductive logic. *Logic Journal of the IGPL*, 14(2):189–219(31), March 2006.

[7] Charles Sanders Peirce. *Collected Papers of Charles Sanders Peirce*. Harvard University Press, 1931–1958. Volumes 1–8, edited by Charles Hartshorne, Paul Weiss and Arthur Burks.

[8] David Poole. Representing knowledge for logic-based diagnosis. In *Proceedings of the International Conference on Fifth Generation Computer Systems*, pages 1282–1290, 1988.

[9] Raymond Reiter and Johan de Kleer. Foundations of assumption-based truth maintenance systems: Preliminary report. In *AAAI*, pages 183–189, 1987.

[10] Murray Shanahan. Prediction is deduction but explanation is abduction. In *Proceedings of IJCAI*, pages 1055–1060, 1989.

[11] Raymond M. Smullyan. *First-Order Logic*. Springer-Verlag, 1968.

[12] G. S. Tseitin. On the complexity of derivations in the propositional calculus. In A. O. Slisenko, editor, *Structures in Constructive Mathematics and Mathematical Logic, Part II*, pages 115–125. Consultants Bureau, New-York-London, 1968. Translated from Russian.