

Generating Schemata of Resolution Proofs

Vincent Aravantinos, Nicolas Peltier
CNRS, LIG/TU Wien

June 2011

Abstract

Two distinct algorithms are presented to extract (schemata of) resolution proofs from closed tableaux for propositional schemata [4]. The first one handles the most efficient version of the tableau calculus but generates very complex derivations (denoted by rather elaborate rewrite systems). The second one has the advantage that much simpler systems can be obtained, however the considered proof procedure is less efficient.

In [2, 4] a tableau calculus (called STAB) is presented for reasoning on schemata of propositional problems. This proof procedure is able to test the validity of logical formulæ built on a set of indexed propositional symbols, using generalized connectives such as $\bigvee_{i=1}^n$ or $\bigwedge_{i=1}^n$, where i, n are part of the language (n denotes a parameter, i.e. an existentially quantified variable). A schema is unsatisfiable iff it is unsatisfiable for every value of n . STAB combines the usual expansion rules of propositional logic with some delayed instantiation schemes that perform a case-analysis on the value of the parameter n . Termination is ensured for a specific class of schemata, called *regular*, thanks to a loop detection rule which is able to prune infinite tableaux into finite ones, by encoding a form a mathematical induction (by “descente infinie”). A related algorithm, called DPLL* and based on an extension of the Davis-Putnam-Logemann-Loveland procedure, is presented in [3].

In the present work, we show that resolution proofs can be automatically extracted from the closed tableaux constructed by STAB or DPLL* on unsatisfiable schemata. More precisely, we present an algorithm that, given a closed tableau \mathcal{T} for a schema ϕ_n , returns a schema of a refutation of ϕ_n in the resolution calculus [9]. In the usual propositional case, it is well-known that algorithms exist to extract resolution proofs from closed tableaux constructed either by the usual structural rules [11, 13] or by the DPLL algorithm [7, 6]. The resolution proofs are used in various applications, for instance for certification [14], for abstraction-refinement [10] or for explanations generation [8]. The present paper extends these techniques to propositional schemata. Beside the previously mentioned applications, this turned out to be particularly important in the context of the ASAP project [1] in which schemata calculi are applied to the formalisation and analysis of mathematical proofs via cut-elimination. Indeed, the algorithm used for cut-elimination, called CERES [5], explicitly relies

on the existence of a resolution proof of the so-called *characteristic clause set* extracted from the initial proof. The cut-free proof is reconstructed from this refutation, by replacing the clauses occurring in this set by some “projections” of the original proof. While STAB and DPLL* are able to detect the unsatisfiability of characteristic clause sets, as such this is completely useless since actually it is known that those sets are *always* unsatisfiable (see Proposition 3.2 in [5]). It is thus essential to be able to generate *explicitly* a representation of the resolution proof. This is precisely the aim of the present paper. Since the initial formula depends on a parameter \mathbf{n} , its proof will also depend on \mathbf{n} (except in very particular and trivial cases), i.e. it must be a *schema of resolution proof* (which will be encoded by recursive definitions).

The rest of the paper is structured as follows. In Section 1 we introduce the basic notions and notations used throughout our work, in particular the logic of *propositional schemata* (syntax and semantics). In Section 2 we define a tableau-based proof procedure for this logic. This calculus simulates both STAB and DPLL* (for the specific class of schemata considered in the present paper). In Section 3 we provide an algorithm to extract resolution proofs from closed tableaux. Similarly to the formulæ themselves, the constructed derivations are represented by rewrite systems. In Section 4 we introduce a second algorithm which generates simpler derivations but that requires that one of the closure rules defined in Section 2 (the so-called *Loop Detection* rule) be replaced by a less powerful rule, called the *Global Loop Detection* rule. Section 5 briefly concludes our work.

1 Propositional schemata

The definitions used in the present paper differ from the previous ones, but the considered logic is equivalent to the class of regular schemata considered in [2] (it is thus strictly less expressive than general schemata, for which the satisfiability problem is undecidable). We consider three disjoint sets of symbols: a set of *arithmetic variables* \mathcal{V} , a set of *propositional variables* Ω and a set of *defined symbols* Υ . Let \prec be a total well-founded ordering on the symbols in Υ . An *index expression* is either a natural number or of the form $\mathbf{n} + k$, where \mathbf{n} is an arithmetic variable and k is a natural number. Let I be a set of index expressions. The set $\mathcal{F}(I)$ of *formulæ built on I* is inductively defined as follows: if $p \in \Omega \cup \Upsilon$ and $\alpha \in I$ then $p_\alpha \in \mathcal{F}(I)$; $\top, \perp \in \mathcal{F}(I)$; and if $\phi, \psi \in \mathcal{F}(I)$ then $\neg\phi, \phi \vee \psi, \phi \wedge \psi, \phi \Rightarrow \psi$ and $\phi \Leftrightarrow \psi$ are in $\mathcal{F}(I)$.

Definition 1 We assume that each element $v \in \Upsilon$ is mapped to two rewrite rules ρ_v^1 and ρ_v^0 that are respectively of the form $v_{\mathbf{i}+1} \rightarrow \phi$ (inductive case) and $v_0 \rightarrow \psi$ (base case), where $\phi \in \mathcal{F}(\{\mathbf{i} + 1, \mathbf{i}, 0\})$, $\psi \in \mathcal{F}(\{0\})$ and:

1. For every atom τ_α occurring in ϕ such that $\tau \in \Upsilon$ we have either $\tau \prec v$ and $\alpha \in \{\mathbf{i} + 1, \mathbf{i}, 0\}$ or $\tau = v$ and $\alpha \in \{0, \mathbf{i}\}$.
2. For every atom τ_α occurring in ψ such that $\tau \in \Upsilon$ we have $\tau \prec v$ and $\alpha = 0$. ◇

We denote by \mathcal{R} the rewrite system: $\{\rho_v^1, \rho_v^0 \mid v \in \Upsilon\}$. The rules ρ_v^1 and ρ_v^0 are provided by the user, they encode the semantics of the defined symbols.

Proposition 2 \mathcal{R} is convergent.

PROOF. By Conditions 1 and 2 in Definition 1, the rules in \mathcal{R} either strictly decrease the values of the defined symbols occurring in the formula w.r.t. \prec or do not increase the value of these symbols but strictly decreases the value of their indices. Thus termination is obvious. Confluence is then immediate since the system is orthogonal. ■

For every formula ϕ , we denote by $\phi \downarrow_{\mathcal{R}}$ the unique normal form of ϕ .

A *schema* (of parameter \mathbf{n}) is an element of $\mathcal{F}(\{0, \mathbf{n}, \mathbf{n} + 1\})$. We denote by $\phi\{\mathbf{n} \leftarrow k\}$ the formula obtained from ϕ by replacing every occurrence of \mathbf{n} by k . Obviously for any schema ϕ , $\phi\{\mathbf{n} \leftarrow k\} \in \mathcal{F}(\{0, k, k + 1\})$. A *propositional formula* is a formula $\phi \in \mathcal{F}(\mathbb{N})$ containing no defined symbols. Notice that if $\phi \in \mathcal{F}(\mathbb{N})$ then $\phi \downarrow_{\mathcal{R}}$ is a propositional formula.

Proposition 3 If $\phi \in \mathcal{F}(\mathbb{N})$ then $\phi \downarrow_{\mathcal{R}}$ is a propositional formula.

PROOF. By definition of \mathcal{R} , $\phi \downarrow_{\mathcal{R}} \in \mathcal{F}(\mathbb{N})$. Furthermore, if $\phi \downarrow_{\mathcal{R}}$ contains a defined symbol v then either ρ_v^1 or ρ_v^0 applies, which is impossible. ■

An *interpretation* is a function mapping every arithmetic variable \mathbf{n} to a natural number and every atom of the form p_k (where $k \in \mathbb{N}$) to a truth value true or false. An interpretation I *validates* a propositional formula ϕ iff one of the following conditions holds: ϕ is of the form p_k and $I(p_k) = \text{true}$; ϕ is of the form $\neg\psi$ and I does not validate ψ ; or ϕ is of the form $\psi_1 \vee \psi_2$ (resp. $\psi_1 \wedge \psi_2$) and I validates ψ_1 or ψ_2 (resp. ψ_1 and ψ_2). I *validates a schema* ϕ (written $I \models \phi$) iff I validates $\phi\{\mathbf{n} \leftarrow I(\mathbf{n})\} \downarrow_{\mathcal{R}}$. We write $\phi \models \psi$ if every interpretation I validating ϕ also validates ψ and $\phi \equiv \psi$ if $\phi \models \psi$ and $\psi \models \phi$.

Example 4 The schema $p_0 \wedge \bigwedge_{i=1}^{\mathbf{n}} (p_{i-1} \Rightarrow p_i) \wedge \neg p_{\mathbf{n}}$ is encoded by $p_0 \wedge v_{\mathbf{n}} \wedge \neg p_{\mathbf{n}}$, where v is defined by the rules: $v_{i+1} \rightarrow (\neg p_i \vee p_{i+1}) \wedge v_i$ and $v_0 \rightarrow \top$.

The schema $\bigvee_{i=1}^{\mathbf{n}} p_i \wedge \bigwedge_{i=1}^{\mathbf{n}} \neg p_i$ is encoded by $\tau_{\mathbf{n}} \wedge \tau'_{\mathbf{n}}$, where τ and τ' are defined by the rules: $\tau_{i+1} \rightarrow p_{i+1} \vee \tau_i$, $\tau_0 \rightarrow \perp$, $\tau'_{i+1} \rightarrow \neg p_{i+1} \wedge \tau'_i$ and $\tau'_0 \rightarrow \top$.

Both schemata are obviously unsatisfiable.

The schema $(p_{\mathbf{n}} \Leftrightarrow (p_{\mathbf{n}-1} \Leftrightarrow (\dots (p_1 \Leftrightarrow p_0) \dots)))$ is defined by $v'_{\mathbf{n}}$, where: $v'_{i+1} \rightarrow (p_{i+1} \Leftrightarrow v'_i)$ and $v'_0 \rightarrow p_0$. ♣

2 Proof procedure

In this section we define the proof procedure used to decide the validity of propositional schemata. We assume for simplicity that the considered schemata are in negative normal form and that the defined symbols occur only positively¹.

¹If a defined symbol v occurs negatively then it is easy to replace every literal of the form $\neg v_{\alpha}$ by an atom \overline{v}_{α} where \overline{v} denotes the complementary of v . The rewrite rules for \overline{v} are obtained by negating the right-hand side of the rules of v , e.g. the atom \overline{v} corresponding to the symbol v in Example 4 is defined by the rewrite rules $\overline{v}_{i+1} \rightarrow (p_i \wedge \neg p_{i+1}) \vee \overline{v}_i$ and $\overline{v}_0 \rightarrow \perp$.

The procedure is similar to the one presented in [2] and based on propositional block tableaux [12]. It constructs a tree labeled by finite sets of schemata,

using *expansion rules* of the form: $\frac{\Phi}{\Psi_1 \mid \dots \mid \Psi_k}$, meaning that a leaf whose label is of the form $\Phi \cup \Phi'$ (and does not already contain \perp) is expanded by adding k children labeled by $\Phi' \cup \Psi_1, \dots, \Phi' \cup \Psi_k$ respectively. If α is a node in \mathcal{T} , then $\mathcal{T}(\alpha)$ denotes the label of α . The expansion rules are defined as follows:

$$\text{Normalisation: } \frac{v_\alpha}{v_\alpha \downarrow \mathcal{R}} \quad \text{if } v_\alpha \text{ is reducible w.r.t. } \mathcal{R}$$

$$\begin{array}{ccc} \vee\text{-Decomposition} & \wedge\text{-Decomposition} & \text{Closure} \\ \frac{\phi \vee \psi}{\phi \mid \psi} & \frac{\phi \wedge \psi}{\phi, \psi} & \frac{\phi, \neg\phi}{\perp} \end{array}$$

$$\text{Purity rule: } \frac{p_{n+k}}{\top} \quad \frac{\neg p_{n+k}}{\top} \quad \text{if } k > 0 \text{ and the previous rules do not apply}$$

Note that the notion of pure literal is much simpler here than in [2]. This is due to the fact that no constant index distinct from 0 and no index of the form $i + k$ where $k > 1$ are allowed.

A node that is irreducible w.r.t. all the previous rules is called a *layer*. The *Loop Detection rule* applies to nodes containing previously generated layers:

$$\text{Loop Detection: } \frac{\Phi}{\perp} \quad \text{if a non leaf layer labeled by } \Phi \text{ exists in the tree}$$

Note that the layer does not necessarily occur in the same branch as the one on which the rule is applied. The essential point is that the set of schemata Φ has already been considered somewhere - consequently if it has a model then an open branch necessarily exists elsewhere in the tree.

Finally, the last rule performs a case analysis on n (in this particular rule, Φ denotes the whole label of the considered node):

$$\text{Explosion: } \frac{\Phi}{\Phi\{n \leftarrow 0\} \mid \Phi\{n \leftarrow n+1\}} \quad \begin{array}{l} \text{if no other rule applies} \\ \text{and } n \text{ occurs in } \Phi \end{array}$$

A tableau is *closed* if the labels of all leaves contain \perp .

Theorem 5 *The tableau expansion rules are terminating, i.e. there is no infinite sequence $(\mathcal{T}_i)_{i \in \mathbb{N}}$ such that for every $i \in \mathbb{N}$, \mathcal{T}_{i+1} is obtained from \mathcal{T}_i by applying one of the previous rules.*

PROOF. The termination of the rules Normalisation, Decomposition, Closure and Loop Detection is obvious: indeed, the Normalisation rule strictly decreases the value of the indices occurring in the formulæ, whereas the other rules cannot increase these indices and strictly reduces the size of the label (i.e. the number of symbols). Thus we only have to show that the number of layers is finite. Let

\mathcal{S} be the set of layers generated by the expansion rule on a given set of schemata of some parameter \mathbf{n} . By definition, a layer is irreducible by the Decomposition rules, thus every formula occurring in $\mathcal{T}(\alpha)$ where $\alpha \in \mathcal{S}$ must be a literal. By irreducibility w.r.t. the Normalisation and Purity rules, the indices of these literals must be either \mathbf{n} or 0. Since \prec is well-founded, and since all labels are finite, the number of symbols occurring in the tableau must be finite, hence the set $\{\mathcal{T}(\alpha) \mid \alpha \in \mathcal{S}\}$ is finite. By the order of application of the expansion rules, the Explosion rule cannot be applied on two layers labeled by the same set of formulæ. Thus \mathcal{S} is finite. ■

The next theorem states that the calculus is correct:

Theorem 6 *If \mathcal{T} contains an irreducible leaf not containing \perp , then the label of the root of \mathcal{T} is satisfiable.*

PROOF. Let α be the root of \mathcal{T} . The proof is by induction on the depth of the irreducible node in \mathcal{T} .

- Assume that α is irreducible. α must be a layer, thus $\mathcal{T}(\alpha)$ is a set of literals, indexed by the parameter \mathbf{n} or by 0 (as shown in the proof of Theorem 5). By irreducibility w.r.t. the Explosion rule, \mathbf{n} cannot occur in the label. Thus $\mathcal{T}(\alpha)$ is a set of literals indexed by 0. Furthermore, $\mathcal{T}(\alpha)$ cannot contain two complementary literals, hence must be satisfiable.
- If α is not irreducible, then some expansion rule must be applied on α . The rule cannot be the Closure rule, nor the Loop Detection rule (otherwise α would necessarily contain \perp). We distinguish several cases:
 - If the \wedge -decomposition rule is applied on α then α has one child β . By the induction hypothesis $\mathcal{T}(\beta)$ is satisfiable. By definition of the rule, we have $\mathcal{T}(\alpha) \equiv \mathcal{T}(\beta)$ hence $\mathcal{T}(\alpha)$ is satisfiable.
 - If the \vee -decomposition rule is applied on α then α has two children β_1 and β_2 . By definition, the irreducible node must occur in the branch of β_1 or β_2 , say β_1 . By the induction hypothesis $\mathcal{T}(\beta_1)$ is satisfiable. By definition of the rule, we have $\mathcal{T}(\beta_1) \models \mathcal{T}(\alpha)$ hence $\mathcal{T}(\alpha)$ is satisfiable.
 - If the Explosion rule is applied on α then α has two children β_1 and β_2 corresponding to the case $\mathbf{n} \leftarrow 0$ and $\mathbf{n} \leftarrow \mathbf{n} + 1$ respectively. By definition, the irreducible node must occur in the branch of β_1 or β_2 . If it occurs in the branch of β_1 , then by the induction hypothesis β_1 has a model I . Moreover, by definition of the rule $\mathcal{T}(\beta_1)$ contains no occurrence of \mathbf{n} (since \mathbf{n} is replaced by 0), thus the truth value of $\mathcal{T}(\beta_1)$ is independent of the value of \mathbf{n} . We may thus assume that $I(\mathbf{n}) = 0$. Then $I \models \mathcal{T}(\alpha)$ iff $I \models \mathcal{T}(\alpha)\{\mathbf{n} \leftarrow 0\}$, i.e. iff $I \models \mathcal{T}(\beta_1)$. Therefore, I is a model of α .
 - If the irreducible node is a descendant of β_2 , then by the induction hypothesis β_2 has a model I . Let J be an interpretation coinciding

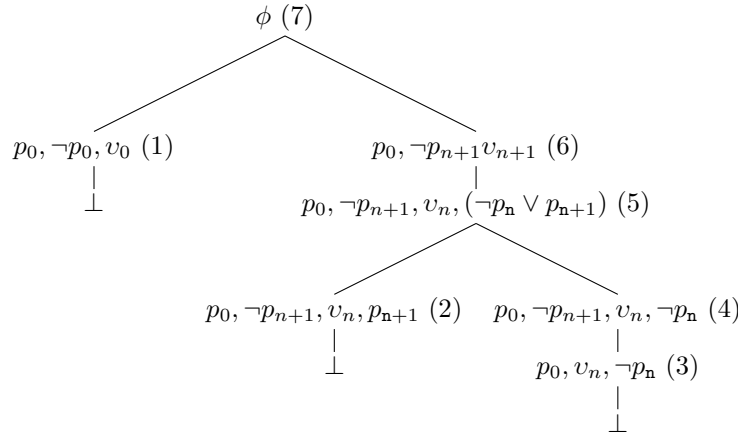
with I except that $J(n) \stackrel{\text{def}}{=} I(n) + 1$. By definition, we have $J \models \mathcal{T}(\alpha)$ iff $I \models \mathcal{T}(\alpha)\{n \leftarrow n + 1\}$, i.e. if $I \models \mathcal{T}(\beta_2)$. Therefore, J is a model of α .

- If the Purity rule is applied on α , then α has one child β . We have $\mathcal{T}(\alpha) = \mathcal{T}(\beta) \cup \{p_{n+k}\}$ (resp. $\mathcal{T}(\beta) \cup \{\neg p_{n+k}\}$), where $k > 0$. By the induction hypothesis, $\mathcal{T}(\beta)$ has a model I . We remark that the truth value of $\mathcal{T}(\beta)$ does not depend on the value of p_{n+k} . Indeed, p_{n+k} cannot occur in $\mathcal{T}(\beta)$ (otherwise the Closure rule would be applicable on $\mathcal{T}(\alpha)$ which is impossible). Furthermore, by irreducibility w.r.t. the Normalisation rule, the indices of the defined symbols occurring in α must be 0 or n . Since the rewrite rules in \mathcal{R} cannot increase the value of these indices, the truth value of these indexed defined symbols depends only of the values of the atoms indexed by $I(n), I(n-1), \dots, I(0)$. Thus we may assume that $I(p_{n+k}) = \text{true}$ (resp. $I(p_{n+k}) = \text{false}$). Hence $I \models \mathcal{T}(\alpha)$.

■

We will *not* prove the converse (namely that the root of every closed tableau is unsatisfiable), because this is subsumed by Theorem 18 in Section 3 (ensuring the existence of a resolution proof for every instance of the root schema).

Example 7 The schema $\phi : p_0 \wedge \neg p_n \wedge v_n$, where v is defined as in Example 4 is unsatisfiable. For instance, $\phi\{n \leftarrow 2\}$ is $p_0 \wedge \neg p_2 \wedge (\neg p_0 \vee p_1) \wedge (\neg p_1 \vee p_2)$. The reader can check that the expansion rules construct the following tableau. The root is actually a layer, hence the Explosion rule is applied on it. The node (3) is deduced by the Purity rule and closed by applying the Loop Detection rule (with the root). The other rule applications are straightforward.



♣

The DPLL* procedure in [3] can be simulated by the previous expansion rules, simply by adding for each propositional symbol $p \in \Omega$, a defined symbol v^p with two rules: $v_{i+1}^p \rightarrow ((p_i \vee \neg p_i) \wedge v_i^p)$ and $v_0^p \rightarrow \top$. Then the case splitting

rule of the DPLL procedure on a variable p corresponds to an application of the \wedge -rule on $v_{n+1}^p \downarrow_{\mathcal{R}}$ (yielding $p_n \vee \neg p_n$) followed by an application of the \vee -rule on $p_n \vee \neg p_n$. The propagation rule is then simulated by combining the \vee -rule and the closure rule².

3 Constructing resolution proofs

3.1 Propositional resolution calculus

We first briefly recall the notion of resolution inference (in propositional logic). A *literal* is either an atom p_k or the negation of an atom $\neg p_k$ (where $p \in \Omega$ and $k \in \mathbb{N}$). A *clause* is a (possibly empty) disjunction (or set) of literals. A *derivation* from a set of clauses S is a finite sequence C_1, \dots, C_m such that for every $i \in [1, m]$, C_i is either an element of S or obtained from C_1, \dots, C_{m-1} by applying the resolution rule, defined as follows:
$$\frac{p_k \vee X \quad \neg p_k \vee Y}{X \vee Y}$$

A *refutation* is a derivation containing \perp (the empty clause). For any formula ϕ , Δ is a *derivation from ϕ* if it is a derivation from a clausal form of ϕ .

It is well-known [9] that every unsatisfiable set of (propositional) clauses has a refutation. In the context of propositional schemata, this means that every instance $\phi\{\mathbf{n} \leftarrow k\} \downarrow_{\mathcal{R}}$ of an unsatisfiable propositional schema ϕ of parameter \mathbf{n} has a refutation Δ_k (which in general depends on k). The problem is then to construct a representation of the sequence of refutations $\Delta_0, \Delta_1, \dots, \Delta_k, \dots$. This sequence may be seen as a *schema of refutation* which (similarly to the semantics of the defined symbols) will be denoted by a *system of rewrite rules*. From now, we assume that the considered schema is in conjunctive normal form (i.e. it contains no conjunctions inside disjunctions, even if these conjunctions are “hidden” in the inductive definitions of the defined symbols, e.g. the schema $p_n \vee v_n$, where v is defined as in Example 4 is forbidden).

3.2 A language for representing refutations

Additional definitions are needed to provide suitable formal languages for denoting such schemata of derivations. Let \mathcal{D} and \mathcal{X} be two disjoint sets of symbols (disjoint from \mathcal{V} , Ω and Υ). The symbols in \mathcal{D} are the Δ -*symbols* and the ones in \mathcal{X} are the Δ -*variables*. The symbols in \mathcal{X} are intended to be instantiated by schemata, whereas the symbols $d \in \mathcal{D}$ will denote schemata of refutations, defined by induction (and possibly depending on an additional argument Δ denoting a formula). We assume that \prec is extended into a well-founded ordering on \mathcal{D} .

Formally, the set of Δ -expressions is inductively defined as follows:

- All schemata and all Δ -variables are Δ -expressions.

²This “trick” does not actually simulate the full procedure in [3], because the latter handles schemata that are more complex than the ones considered in the present paper, possibly containing nested iterations.

- If $d \in \mathcal{D}$, α is an index expression and Δ is a Δ -expression, then d_α and $d_\alpha(\Delta)$ are Δ -expressions.
- If Δ and Γ are Δ -expressions then $\Delta \vee \Gamma$, $\Delta \wedge \Gamma$ and $\Delta \cdot \Gamma$ are Δ -expressions.

The expression $\Delta \cdot \Gamma$ is to be interpreted as the concatenation of two sequences Δ and Γ . Note that Δ -expressions can represent uniformly schemata of clauses, schemata of clause sets, or schemata of derivations (i.e. schemata of sequences of clauses). For the sake of conciseness and simplicity, the previous definition does *not* ensure that the constructions are well-typed, e.g. we can consider Δ -expressions of the form $\Delta \vee \Gamma$ where Δ and Γ are two *sequences of clauses* (which obviously does not make sense: Δ and Γ should rather be *clauses*). But in the forthcoming definitions we will ensure that all the considered Δ -expressions are well-typed.

Example 8 Let $d \in \mathcal{D}$. Then $(p_2 \vee q_0) \cdot d_2(q_0) \cdot \neg q_0 \cdot \perp$ is a Δ -expression. ♣

A Δ -expression is *ground* if it contains no index variable and no Δ -variable. In order to interpret (ground) Δ -expressions, the value of the Δ -symbols is specified using a rewrite system, exactly as schemata can be transformed into propositional formulæ by interpreting the defined symbols (using the rewrite system \mathcal{R}). The rewrite systems used in this section are more complicated than in the previous one, since the symbols in \mathcal{D} may have an additional argument.

A Δ -*substitution* is a function mapping every arithmetic variable to an index expression and every Δ -variable to a Δ -expression. If Δ is a Δ -expression and σ is a Δ -substitution, then $\Delta\sigma$ denotes the Δ -expression obtained from Δ by replacing every variable $x \in \mathcal{V} \cup \mathcal{X}$ by $\sigma(x)$.

Definition 9 A \mathcal{D} -*system* is a set of *rewrite rules* of the form $\Delta \rightarrow \Gamma$, where Δ, Γ are two Δ -expressions such that every arithmetic variable and every Δ -variable occurring in Γ also occurs in Δ . A \mathcal{D} -system is *propositional* if it contains no Δ -variables (it may contain arithmetic variables).

Given two Δ -expressions Δ and Γ and a \mathcal{D} -system \mathfrak{R} , we write $\Delta \rightarrow_{\mathfrak{R}} \Gamma$ if there exists a rule $\Delta' \rightarrow \Gamma'$ in \mathfrak{R} and a Δ -substitution σ such that Γ is obtained from Δ by replacing an occurrence of an expression $\Delta'\sigma$ by $\Gamma'\sigma$. ◇

For matching, the associativity and commutativity of logical symbols are not taken into account in general, *except for conjunctions occurring at the root level* (this rather unusual convention is needed to ensure confluence without having to bother on the order of the schemata at the root level). For instance the rule $d(p \wedge ((r \wedge q) \vee \neg r)) \rightarrow p$ does not apply on $d(p \wedge (\neg r \vee (r \wedge q)))$ nor on $d(p \wedge ((q \wedge r) \vee \neg r))$, but it applies on $d(((r \wedge q) \vee \neg r) \wedge p)$. Similarly, $d(p \wedge q) \rightarrow p$ applies on $d(p)$ by assuming $q = \top$.

Example 10 Consider the following rewrite system (Z is a Δ -variable).

$$\{d_{i+1}(Z) \rightarrow (\neg p_{i+1} \vee p_i) \cdot (p_i \vee Z) \cdot d_i(Z), \quad d_0(Z) \rightarrow \neg p_0 \cdot Z\}$$

The reader can check that it reduces the Δ -expression of Example 8 to:

$$(p_2 \vee q_0) \cdot (\neg p_2 \vee p_1) \cdot (p_1 \vee q_0) \cdot (\neg p_1 \vee p_0) \cdot (p_0 \vee q_0) \cdot \neg p_0 \cdot q_0 \cdot \neg q_0 \cdot \perp$$

This last expression is a refutation. ♣

3.3 From closed tableaux to resolution proofs

Let \mathcal{T} be a closed tableau of a schema ϕ . The general idea is to construct, from \mathcal{T} , a \mathcal{D} -system $\mathfrak{R}(\mathcal{T})$ representing a schema of refutation for ϕ . Obviously, $\mathfrak{R}(\mathcal{T})$ represents an inductive proof of the assertion: “for every $\mathbf{n} \in \mathbb{N}$, the corresponding instance of ϕ is unsatisfiable”. Ideally, we would just refute the base case, and then build a refutation of ϕ at $\mathbf{n} + 1$ from a refutation of ϕ at \mathbf{n} . However, as often in inductive reasoning, we need to generalize the conjecture in order to refute it properly. This is done as follows: recall that our aim is to construct a refutation of ϕ , i.e. a derivation of \perp from ϕ ; instead, however, $\mathfrak{R}(\mathcal{T})$ will describe how to build a derivation of X from $\phi \vee X$, for *any* X (formally, X will be a Δ -variable). Then, our original goal will be reached by just substituting \perp to X . In practice, we need to generalize even more this reasoning since the construction of $\mathfrak{R}(\mathcal{T})$ is done by mapping *every node* α of \mathcal{T} to some rewrite rules. So, instead of considering only the root schema ϕ , we need to consider all the formulæ $\{\phi_1, \dots, \phi_k\}$ that occur in $T(\alpha)$. And, instead of building a derivation of X from $\phi \vee X$, we build a derivation of $X_1 \vee \dots \vee X_k$ from $(\phi_1 \vee X_1) \wedge \dots \wedge (\phi_k \vee X_k)$, for some Δ -variables X_1, \dots, X_k . More precisely we build a derivation of a clause $C \subseteq X_1 \vee \dots \vee X_k$, since some formulæ $\phi_i \vee X_i$ may be useless. We retrieve our original goal when we just substitute the root of \mathcal{T} to α .

The following definition constructs a \mathcal{D} -system $\mathfrak{R}(\mathcal{T})$ and two Δ -symbols ν^α and μ^α such that, if $\mathcal{T}(\alpha) = \{\phi_1, \dots, \phi_k\}$ and U denotes the formula $(\phi_1 \vee X_1) \wedge \dots \wedge (\phi_k \vee X_k)$ then $\mu_n^\alpha(U)$ denotes the above clause C and $\nu_n^\alpha(U)$ denotes a derivation of C from U . This system is constructed by induction on the tableau.

Definition 11 Let \mathcal{T} be a tableau. We map every node α in \mathcal{T} to two Δ -symbols ν^α and μ^α . We assume that the symbols ν^α and μ^α are pairwise distinct. The system of rules $\mathfrak{R}(\mathcal{T})$ is defined by the rules in \mathcal{R} and the following rules, for every node α in \mathcal{T} (we distinguish several cases, according to the rule applied on α):

- If no rule is applied on α : $\nu_n^\alpha((\perp \vee X) \wedge Y) \rightarrow X \quad \mu_n^\alpha((\perp \vee X) \wedge Y) \rightarrow X$
- If the Normalisation rule is applied on α , using a formula ϕ , yielding a node β :

$$\begin{aligned} \nu_n^\alpha((\phi \vee X) \wedge Y) &\rightarrow \nu_n^\beta((\phi \downarrow_{\mathcal{R}} \vee X) \wedge Y) \\ \mu_n^\alpha((\phi \vee X) \wedge Y) &\rightarrow \mu_n^\beta((\phi \downarrow_{\mathcal{R}} \vee X) \wedge Y) \end{aligned}$$

- If the Closure rule is applied on α , using ϕ and $\neg\phi$:

$$\nu_n^\alpha((\phi \vee X) \wedge (\neg\phi \vee Y) \wedge Z) \rightarrow (\neg\phi \vee Y) \cdot (\phi \vee X) \cdot (X \vee Y)$$

$$\mu_n^\alpha((\phi \vee X) \wedge (\neg\phi \vee Y) \wedge Z) \rightarrow (X \vee Y)$$

- If \wedge -Decomposition is applied on α , yielding a child β :

$$\nu_n^\alpha(((\phi_1 \wedge \phi_2) \vee X) \wedge Y) \rightarrow \nu_n^\beta((\phi_1 \vee X) \wedge (\phi_2 \vee X) \wedge Y)$$

$$\mu_n^\alpha(((\phi_1 \wedge \phi_2) \vee X) \wedge Y) \rightarrow \mu_n^\beta((\phi_1 \vee X) \wedge (\phi_2 \vee X) \wedge Y)$$

- If \vee -Decomposition is applied on α using a formula $\phi \vee \psi$ and yielding two children β_1 and β_2 :

$$\nu_n^\alpha(((\phi_1 \vee \phi_2) \vee X) \wedge Y) \rightarrow \nu_n^{\beta_1}((\phi_1 \vee (\phi_2 \vee X)) \wedge Y) \cdot \nu_n^{\beta_2}(\mu_n^{\beta_1}((\phi_1 \vee (\phi_2 \vee X)) \wedge Y) \wedge Y)$$

$$\mu_n^\alpha(((\phi_1 \vee \phi_2) \vee X) \wedge Y) \rightarrow \mu_n^{\beta_2}(\mu_n^{\beta_1}((\phi_1 \vee (\phi_2 \vee X)) \wedge Y) \wedge Y)$$

- If the Purity rule is applied on α , on a formula ϕ , yielding a node β :

$$\nu_n^\alpha((\phi \vee X) \wedge Y) \rightarrow \nu_n^\beta(Y) \quad \mu_n^\alpha((\phi \vee X) \wedge Y) \rightarrow \mu_n^\beta(Y)$$

- If the Loop Detection rule is applied on α , using a layer β :

$$\nu_n^\alpha(X) \rightarrow \nu_n^\beta(X) \quad \mu_n^\alpha(X) \rightarrow \mu_n^\alpha(X)$$

- If the Explosion rule is applied on α , yielding two children β_1 and β_2 , corresponding to the cases $n \leftarrow 0$ and $n \leftarrow n + 1$ respectively:

$$\nu_0^\alpha(X) \rightarrow \nu_0^{\beta_1}(X) \quad \nu_{n+1}^\alpha(X) \rightarrow \nu_n^{\beta_1}(X) \quad \mu_0^\alpha(X) \rightarrow \mu_0^{\beta_2}(X) \quad \mu_{n+1}^\alpha(X) \rightarrow \mu_n^{\beta_2}(X)$$

◇

Note that all the symbols ϕ, ϕ_1, ϕ_2 denote meta-variables, and not Δ -variables (hence they cannot be instantiated during rewriting, in contrast to X, Y, \dots).

Before establishing the properties of $\mathfrak{R}(\mathcal{T})$, we show an example of application:

Example 12 Consider the proof tree of Example 7. The reader can check that $\mathfrak{R}(\mathcal{T})$ contains the following rules:

$$\begin{array}{ll}
\nu_n^1((p_0 \vee X) \wedge (\neg p_0 \vee Y) \wedge Z) & \rightarrow (p_0 \vee X) \cdot (\neg p_0 \vee Y) \cdot (X \vee Y) \\
\mu_n^1((p_0 \vee X) \wedge (\neg p_0 \vee Y) \wedge Z) & \rightarrow X \vee Y \\
\nu_n^2((p_{n+1} \vee X) \wedge (\neg p_{n+1} \vee Y) \wedge Z) & \rightarrow (p_{n+1} \vee X) \cdot (\neg p_{n+1} \vee Y) \cdot (X \vee Y) \\
\mu_n^2((p_{n+1} \vee X) \wedge (\neg p_{n+1} \vee Y) \wedge Z) & \rightarrow X \vee Y \\
\nu_n^3(X) & \rightarrow \nu_n^7(X) \\
\mu_n^3(X) & \rightarrow \mu_n^7(X) \\
\nu_n^4((\neg p_{n+1} \vee X) \wedge Y) & \rightarrow \nu_n^3(Y) \\
\mu_n^4((\neg p_{n+1} \vee X) \wedge Y) & \rightarrow \mu_n^3(Y) \\
\nu_n^5(((\neg p_n \vee p_{n+1}) \vee X) \wedge Y) & \rightarrow \nu_n^2(((p_{n+1}) \vee (\neg p_n \vee X)) \wedge Y) \\
& \quad \cdot \nu_n^4(\mu_n^2((p_{n+1} \vee (\neg p_n \vee X)) \wedge Y) \wedge Y) \\
\mu_n^5(((\neg p_n \vee p_{n+1}) \vee X) \wedge Y) & \rightarrow \mu_n^4(\mu_n^2((p_{n+1} \vee (\neg p_n \vee X)) \wedge Y) \wedge Y) \\
\nu_n^6((v_{n+1} \vee X) \wedge Y) & \rightarrow \nu_n^5(((\neg p_n \vee p_{n+1}) \vee X) \wedge v_n \wedge Y) \\
\mu_n^6((v_{n+1} \vee X) \wedge Y) & \rightarrow \mu_n^5(((\neg p_n \vee p_{n+1}) \vee X) \wedge v_n \wedge Y) \\
\nu_0^7(X) & \rightarrow \nu_0^1(X) \\
\mu_0^7(X) & \rightarrow \mu_0^1(X) \\
\nu_{n+1}^7(X) & \rightarrow \nu_n^6(X) \\
\mu_{n+1}^7(X) & \rightarrow \mu_n^6(X)
\end{array}$$

The Δ -expression $\nu_n^7((p_0 \vee \perp) \wedge (\neg p_n \vee \perp) \wedge (v_n \vee \perp))$ denotes a refutation of $p_0 \wedge \neg p_n \wedge v_n$. This rewrite system is complex and hardly readable, fortunately it can be simplified by instantiating the arguments when possible and by *statically* evaluating the derivations that do not depend on the value of the parameter n . For instance the Δ -symbol ν_n^7 is only called on the formula $T_n = (p_0 \vee \perp) \wedge (\neg p_n \vee \perp) \wedge (v_n \vee \perp)$. Thus the rule $\nu_n^7(X) \rightarrow \nu_0^1(X)$ may be simplified by instantiating X by T_0 and evaluating the right-hand side: $\nu_0^7(T_0) \rightarrow p_0 \cdot \neg p_0 \cdot \perp$.

Similarly, the rule $\nu_{n+1}^7(X) \rightarrow \nu_n^6(X)$ can be replaced by the following rule (in this case only a partial evaluation is possible since some parts of the derivation depend on the value of n): $\nu_{n+1}^7(T_{n+1}) \rightarrow (\neg p_n \vee p_{n+1}) \cdot \neg p_{n+1} \cdot \neg p_n \cdot \nu_n^7(T_n)$.

The obtained system (only containing the two previous rules) is obviously much simpler than the original one, in particular it is propositional (no schema variables occur in it). To improve readability, the expression $\nu_n^7(T_n)$ could be simply replaced by a fresh symbol $\nu_n^{7'}$ (with no argument). ♣

We define the following relation $\prec_{\mathcal{T}}$ on the nodes in a tableau \mathcal{T} .

Definition 13 Let \mathcal{T} be a tableau. $\prec_{\mathcal{T}}$ is the least transitive relation such that $\alpha \prec_{\mathcal{T}} \beta$ if one of the following conditions hold:

1. Either α is a child of β , but α does not correspond to the “ $n \leftarrow n+1$ ” branch of an Explosion rule. This is written $\alpha \prec_{\mathcal{T}}^1 \beta$.
2. Or the Loop Detection rule has been applied on the node β , using the layer α . This is written $\alpha \prec_{\mathcal{T}}^2 \beta$. ◇

Proposition 14 Let \mathcal{T} be a tableau. $\prec_{\mathcal{T}}$ is a strict partial order.

PROOF. By definition, \mathcal{T} has been obtained by a sequence of application of the Expansion rules in Section 2. If α and β are two non-leaf nodes in \mathcal{T} , we write $\alpha \triangleleft \beta$ if the expansion rule on α has been applied before the one of β during this derivation, in chronological order (of course several derivations are possible, we choose one of them arbitrarily). \triangleleft is obviously an ordering. Furthermore, if we have $\alpha \prec_{\mathcal{T}}^2 \beta$ then by the application condition of the Loop Detection rule we must have $\alpha \triangleleft \beta$, since when the rule is applied on β the node α cannot be a leaf, thus an expansion rule must already have been applied on it.

Assume that $\prec_{\mathcal{T}}$ is not an ordering. By definition $\prec_{\mathcal{T}}$ is transitive, thus it must be reflexive, i.e. there is a node α such that $\alpha \prec_{\mathcal{T}} \alpha$. By definition of $\prec_{\mathcal{T}}$ this means that there exists a sequence of nodes β_1, \dots, β_k such that $\beta_1 = \beta_k = \alpha$ and for every $i \in [1, k-1]$, $\beta_i \prec_{\mathcal{T}}^{\epsilon} \beta_{i+1}$ (with $\epsilon = 1, 2$). If for every $i \in [1, k-1]$ we have $\beta_i \prec_{\mathcal{T}}^1 \beta_{i+1}$ then for all $i \in [1, k-1]$ β_i is a child of β_{i+1} which implies that there is a (non trivial) path in the tableau from α to α . This is impossible. Thus there is at least one node β_{i+1} such that the Loop Detection rule is applied on β_{i+1} . W.l.o.g. we can assume that $i+1 = k$. If for every $i \in [1, k-1]$ we have $\beta_i \prec_{\mathcal{T}}^2 \beta_{i+1}$ we have $\beta_i \triangleleft \beta_{i+1}$, hence by transitivity $\alpha \triangleleft \alpha$, which is impossible. Let j the greatest index in $[1, k-1]$ such that $\beta_j \not\prec_{\mathcal{T}}^2 \beta_{j+1}$. We have $\beta_j \prec_{\mathcal{T}}^1 \beta_{j+1} \prec_{\mathcal{T}}^2 \beta_{j+2} \prec_{\mathcal{T}}^2 \dots \prec_{\mathcal{T}}^2 \beta_k$.

Since $\beta_{j+1} \prec_{\mathcal{T}}^2 \beta_{j+2}$, β_{j+1} must be a layer, thus the only rule that can be applied on β_{j+1} is the Explosion rule. Since $\beta_j \prec_{\mathcal{T}}^1 \beta_{j+1}$ β_j cannot correspond to the branch $\mathbf{n} \leftarrow \mathbf{n} + 1$ of the Explosion rule. Thus it corresponds to the branch $\mathbf{n} \leftarrow 0$. But then the nodes $\beta_j, \beta_{j-1}, \dots, \beta_1$ cannot possibly contain \mathbf{n} (since no rule can introduce an occurrence of \mathbf{n} in the tableau, and since by the application condition, the Loop Detection rule cannot be applied between a leaf not containing \mathbf{n} and a layer containing \mathbf{n}). Since $\beta_1 = \beta_k$ this means that $\beta_k, \dots, \beta_{j+1}$ contains no occurrence of \mathbf{n} . But in this case the Explosion rule cannot be applied on β_{j+1} , a contradiction. ■

Lemma 15 *Let \mathcal{T} be a tableau. $\mathfrak{R}(\mathcal{T})$ is convergent.*

PROOF. We extend the ordering $\prec_{\mathcal{T}}$ to the Δ -symbols as follows: $\nu^{\alpha} \prec_{\mathcal{T}} \nu^{\beta}$ and $\mu^{\alpha} \prec_{\mathcal{T}} \mu^{\beta}$ if $\alpha \prec_{\mathcal{T}} \beta$. By definition of $\prec_{\mathcal{T}}$, it is easy to check that all the rules above – except the $\mathbf{n} + 1$ -rewrite rule corresponding to the Explosion rule – strictly decrease the value of the symbols ν^{α} and μ^{α} . Furthermore, they do not increase the value of the indices. The Explosion rule may increase the value of these symbols but strictly decreases their indices. Thus termination is obvious. Confluence is immediate: indeed, since each node is labeled by a set (and not a multiset), the system is necessarily orthogonal (note that we assume that the semantic properties of the logical connectives are not taken into account for the matching, except the AC-properties of the \wedge occurring at root level). ■

For any Δ -expression T , we denote by $T \downarrow_{\mathfrak{R}(\mathcal{T})}$ the normal form of T . We now state the soundness of our algorithm.

Lemma 16 states that the rewrite system $\mathfrak{R}(\mathcal{T})$ indeed fulfils the desired property.

Lemma 16 *Let \mathcal{T} be a closed tableau. Let α be a node in \mathcal{T} . Let $k \in \mathbb{N}$. Let $\mathcal{T}(\alpha) = \{\phi_1, \dots, \phi_n\}$. Let X_1, \dots, X_n be a set of pairwise distinct variables in \mathcal{V} . Let $U = (\phi_1 \vee X_1) \wedge \dots \wedge (\phi_n \vee X_n)$. Then $\nu_k^\alpha(U) \downarrow_{\mathfrak{R}(\mathcal{T})}$ is a derivation from $U \downarrow_{\mathcal{R}}$ of $\mu^\alpha(U) \downarrow_{\mathfrak{R}(\mathcal{T})}$.*

PROOF. The proof is by induction on the pair (α, k) , using the lexicographic extension of the ordering $\prec_{\mathcal{T}}$ on the nodes in \mathcal{T} and of the usual ordering on natural numbers (this ordering is obviously well-founded since \mathcal{T} is finite). We distinguish several cases, according to the expansion rule that is applied on α .

- If no rule is applied on α then $\mathcal{T}(\alpha)$ must contain \perp . W.l.o.g., we assume that $\phi_1 = \perp$. By Definition 11, we have $\nu_k^\alpha(U) \downarrow_{\mathfrak{R}(\mathcal{T})} = X_1$ and $\mu^\alpha(U) \downarrow_{\mathfrak{R}(\mathcal{T})} = X_1$, thus the proof is immediate (since X_1 is obviously a derivation of X_1).
- If the closure rule is applied on α then $\mathcal{T}(\alpha)$ must contain two schemata ψ and $\neg\psi$. W.l.o.g., we assume that $\phi_1 = \psi$ and $\phi_2 = \neg\psi$. By Definition 11, we have $\nu_k^\alpha(U) \downarrow_{\mathfrak{R}(\mathcal{T})} = (\psi \vee X_1) \cdot (\neg\psi \vee X_2) \cdot (X_1 \vee X_2)$ and $\mu^\alpha(U) \downarrow_{\mathfrak{R}(\mathcal{T})} = (X_1 \vee X_2)$ hence the proof is completed, since $(\psi \vee X_1) \cdot (\neg\psi \vee X_2) \cdot (X_1 \vee X_2)$ is a derivation of $X_1 \vee X_2$.
- Assume that \wedge -Decomposition is applied on a schema $\psi_1 \wedge \psi_2$. W.l.o.g., we assume that $\phi_1 = (\psi_1 \wedge \psi_2)$. Let β be the child of α . Let $U' = (\phi_2 \vee X_2) \wedge \dots \wedge (\phi_n \vee X_n)$, i.e. we have $U = ((\psi_1 \wedge \psi_2) \vee X_1) \wedge U'$. By Definition 11, we have $\nu_k^\alpha(U) \downarrow_{\mathfrak{R}(\mathcal{T})} = \nu_k^\beta((\psi_1 \vee X_1) \wedge (\psi_2 \wedge X_2) \wedge U') \downarrow_{\mathfrak{R}(\mathcal{T})}$ and $\mu^\alpha(U) \downarrow_{\mathfrak{R}(\mathcal{T})} = \mu^\beta((\psi_1 \vee X_1) \wedge (\psi_2 \wedge X_2) \wedge U') \downarrow_{\mathfrak{R}(\mathcal{T})}$. Thus, by the induction hypothesis, $\nu_k^\beta((\psi_1 \vee X_1) \wedge (\psi_2 \vee X_2) \wedge U) \downarrow_{\mathfrak{R}(\mathcal{T})}$ is a derivation from $(\psi_1 \vee X_1) \wedge (\psi_2 \wedge X_2) \wedge U' \downarrow_{\mathcal{R}}$ of $\mu^\beta(U) \downarrow_{\mathfrak{R}(\mathcal{T})} = \mu^\alpha(U) \downarrow_{\mathfrak{R}(\mathcal{T})}$. Hence it is also a derivation from $U \downarrow_{\mathcal{R}}$ since U and $(\psi_1 \vee X_1) \wedge (\psi_2 \wedge X_2) \wedge U'$ share the same clausal forms.
- Assume that \vee -Decomposition is applied on a schema $\psi_1 \vee \psi_2$. W.l.o.g., we assume that $\phi_1 = (\psi_1 \vee \psi_2)$. Let β_1 and β_2 be the children of α (corresponding to the schemata ψ_1 and ψ_2 respectively). Let $U' = (\phi_2 \vee X_2) \wedge \dots \wedge (\phi_n \vee X_n)$, i.e. we have $U = ((\psi_1 \vee \psi_2) \vee X_1) \wedge U'$. By Definition 11, we have $\nu_k^\alpha(U) \downarrow_{\mathfrak{R}(\mathcal{T})} = \nu_k^{\beta_1}(U) \downarrow_{\mathfrak{R}(\mathcal{T})} \cdot \nu_k^{\beta_2}(\mu_k^{\beta_1}(U) \wedge U') \downarrow_{\mathfrak{R}(\mathcal{T})}$ and $\mu_k^\alpha(U) \downarrow_{\mathfrak{R}(\mathcal{T})} = \mu_k^{\beta_2}(\mu_k^{\beta_1}(U) \wedge U') \downarrow_{\mathfrak{R}(\mathcal{T})}$.
By the induction hypothesis, $\nu_k^{\beta_1}(U) \downarrow_{\mathfrak{R}(\mathcal{T})}$ is a derivation from $U \downarrow_{\mathcal{R}}$ of $\mu_k^{\beta_1}(U) \downarrow_{\mathfrak{R}(\mathcal{T})}$. Then, again by the induction hypothesis, $\nu_k^{\beta_2}(\mu_k^{\beta_1}(U) \wedge U') \downarrow_{\mathfrak{R}(\mathcal{T})}$ is a derivation from $\mu_k^{\beta_1}(U) \wedge U' \downarrow_{\mathcal{R}}$ of $\mu_k^{\beta_2}(\mu_k^{\beta_1}(U) \wedge U') \downarrow_{\mathfrak{R}(\mathcal{T})}$ i.e. of $\mu_k^\alpha(U) \downarrow_{\mathfrak{R}(\mathcal{T})}$. Consequently, $\nu_k^\alpha(U) \downarrow_{\mathfrak{R}(\mathcal{T})}$ is a derivation from $U \downarrow_{\mathcal{R}}$ of $\mu_k^\alpha(U) \downarrow_{\mathfrak{R}(\mathcal{T})}$.
- Assume that the Loop Detection rule is applied on α , using a node β . $\mathfrak{R}(\mathcal{T})$ contains the rule $\nu_n^\alpha(X) \rightarrow \nu_n^\beta(X)$ and $\mu_n^\alpha(X) \rightarrow \mu_n^\beta(X)$. Then the proof is straightforward, by the induction hypothesis.

- Assume that the Purity rule is applied on α , yielding a node β . Since $\mathcal{T}(\alpha) \supset \mathcal{T}(\beta)$, the proof is immediate (a derivation from a set S is also a derivation from $S \cup S'$).
- Assume that Explosion is applied on α , yielding two nodes β_1 and β_2 (corresponding respectively to the case $\mathbf{n} \leftarrow 0$ and $\mathbf{n} \leftarrow \mathbf{n} + 1$). We distinguish two cases, according to the value of k .
 - If $k = 0$ then we have $\nu_k^\alpha(U) \downarrow_{\mathfrak{R}(\mathcal{T})} = \nu_0^{\beta_1}(U) \downarrow_{\mathfrak{R}(\mathcal{T})}$ and $\mu_k^\alpha(U) \downarrow_{\mathfrak{R}(\mathcal{T})} = \mu_0^{\beta_1}(U) \downarrow_{\mathfrak{R}(\mathcal{T})}$. By the induction hypothesis, $\nu_0^{\beta_1}(U) \downarrow_{\mathfrak{R}(\mathcal{T})}$ is a derivation from U of $\mu_0^{\beta_1}(U) \downarrow_{\mathfrak{R}(\mathcal{T})}$ hence the proof is completed.
 - If $k > 0$ then $\nu_k^\alpha(U) \downarrow_{\mathfrak{R}(\mathcal{T})} = \nu_{k-1}^{\beta_1}(U) \downarrow_{\mathfrak{R}(\mathcal{T})}$ and $\mu_k^\alpha(U) \downarrow_{\mathfrak{R}(\mathcal{T})} = \mu_{k-1}^{\beta_1}(U) \downarrow_{\mathfrak{R}(\mathcal{T})}$. By the induction hypothesis, $\nu_{k-1}^{\beta_1}(U) \downarrow_{\mathfrak{R}(\mathcal{T})}$ is a derivation from U of $\mu_{k-1}^{\beta_1}(U) \downarrow_{\mathfrak{R}(\mathcal{T})}$ hence the proof is completed. ■

Note that (contrarily to all the other cases) we may have $\beta_1 \succ_{\mathcal{T}} \alpha$, but we are using the induction hypothesis on $\nu_{k-1}^{\beta_1}$. This is possible since $k - 1 < k$.

Furthermore, we have the following:

Lemma 17 *Let \mathcal{T} be a closed tableau. Let α be a node in \mathcal{T} . Let $\mathcal{T}(\alpha) = \{\phi_1, \dots, \phi_n\}$. $\mu^\alpha((\phi_1 \vee X_1) \wedge \dots \wedge (\phi_n \vee X_n)) \downarrow_{\mathfrak{R}(\mathcal{T})}$ is a disjunction of formulæ in X_1, \dots, X_n .*

PROOF. By an immediate induction on μ_k^α . ■

Thus in the case in which $X_1 = \dots X_n = \perp$, $\nu_k^\alpha(\Phi)$ denotes a refutation of $\mathcal{T}(\alpha)$, which entails the following theorem, showing the soundness of our algorithm (and entailing in particular the completeness of the tableau calculus).

Theorem 18 *Let \mathcal{T} be a closed tableau containing a node α . Let \mathbf{n} be the parameter of $\mathcal{T}(\alpha)$. Let $\mathcal{T}(\alpha) = \{\phi_1, \dots, \phi_n\}$ and let $\Phi = (\phi_1 \vee \perp) \wedge \dots \wedge (\phi_n \vee \perp)$.*

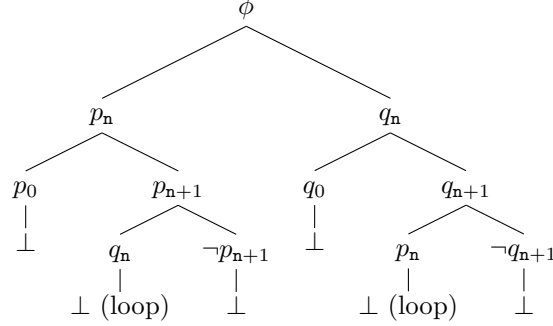
For any $k \in \mathbb{N}$, $\nu_k^\alpha(\Phi\{\mathbf{n} \leftarrow k\}) \downarrow_{\mathfrak{R}(\mathcal{T})}$ is a refutation of $\Phi\{\mathbf{n} \leftarrow k\} \downarrow_{\mathcal{R}}$. Thus $\mathcal{T}(\alpha)$ is unsatisfiable.

PROOF. By Lemma 16, $\nu_k^\alpha(\Phi\{\mathbf{n} \leftarrow k\}) \downarrow_{\mathfrak{R}(\mathcal{T})}$ is a derivation from $\Phi\{\mathbf{n} \leftarrow k\} \downarrow_{\mathcal{R}}$ (hence also from $\Phi\{\mathbf{n} \leftarrow k\} \downarrow_{\mathcal{R}}$) of $\mu_k^\alpha(\Phi\{\mathbf{n} \leftarrow k\}) \downarrow_{\mathfrak{R}(\mathcal{T})}$. By Lemma 17, $\mu_k^\alpha(\Phi\{\mathbf{n} \leftarrow k\}) \downarrow_{\mathfrak{R}(\mathcal{T})} = \perp$. ■

Note that the size of the rewrite system $\mathfrak{R}(\mathcal{T})$ is clearly linear w.r.t. the one of the tableau \mathcal{T} .

The simplification phase used in Example 12 can be applied in a systematic way. However, it is not always sufficient to reduce the rewrite system into a propositional one. Actually, it is not difficult to see that as soon as a node α exists in the tableau on which the \vee -Decomposition rule is applied, yielding two branches β_1 and β_2 that are *both* looping on an ascendant of α , then the use of schema variables cannot be avoided.

Example 19 Consider for instance the schema: $\phi : \neg p_0 \wedge \neg q_0 \wedge (p_n \vee q_n) \wedge v_n$, where v is defined by the rules: $v_{i+1} \rightarrow (q_i \vee \neg p_{i+1}) \wedge (p_i \vee \neg q_{i+1}) \wedge v_i$ and $v_0 \rightarrow \top$. The following tableau is constructed:



The corresponding rewrite system (after partial evaluation and simplification) is the following (ν_n^1 corresponds to the refutation of ϕ):

$$\begin{aligned}
& \nu_n^1(\neg p_0 \wedge \neg q_0 \wedge (p_n \vee q_n) \wedge v_n) \rightarrow \\
& \quad \nu_n^2(\neg p_0 \wedge \neg q_0 \wedge (p_n \vee q_n) \wedge v_n) \cdot \nu_n^3(\neg p_0 \wedge \neg q_0 \wedge q_n \wedge v_n) \\
& \nu_0^2(\neg p_0 \wedge (p_0 \vee X) \wedge Y) \rightarrow \neg p_0 \cdot (p_0 \vee X) \cdot X \\
& \nu_{n+1}^2(\neg p_0 \wedge \neg q_0 \wedge (p_{n+1} \vee X) \wedge v_{n+1}) \rightarrow \\
& \quad (p_{n+1} \vee X) \cdot (q_n \vee \neg p_{n+1}) \cdot (q_n \vee X) \cdot \nu_n^3(\neg p_0 \wedge \neg q_0 \wedge (q_n \vee X) \wedge v_n) \\
& \nu_0^3(\neg p_0 \wedge (q_0 \vee X) \wedge Y) \rightarrow \neg q_0 \cdot (q_0 \vee X) \cdot X \\
& \nu_{n+1}^3(\neg p_0 \wedge \neg q_0 \wedge (q_{n+1} \vee X) \wedge v_{n+1}) \rightarrow \\
& \quad (q_{n+1} \vee X) \cdot (p_n \vee \neg q_{n+1}) \cdot (p_n \vee X) \cdot \nu_n^2(\neg p_0 \wedge \neg q_0 \wedge (p_n \vee X) \wedge v_n)
\end{aligned}$$

The system still contains Δ -variables, although some of them have been removed by static evaluation. Note that it could be further simplified (for instance by moving the axioms such as $\neg p_0$ outside the inductive definitions), but the use of Δ -variables cannot be avoided. ♣

We now focus on an alternative approach that has the advantage that only propositional rewrite systems are generated.

4 Globally looping tableaux

Compared to the previous approach, the second algorithm generates much simpler rewrite systems, but it has the drawback that a more restrictive version of the Loop Detection rule must be used to prune the tableaux into finite ones. At a very high and informal level: in the first approach, we were building mutually inductive proofs of several lemmata, whereas, in the second approach, we manage to have one single invariant proved by a single induction.

We first need to introduce some additional terminology. A node α is of rank k in a tableau \mathcal{T} of root β if there are *exactly* k applications of the Explosion rule between β and α (including β , but not α). $\text{Leaves}(\mathcal{T}, \alpha)$ denotes the set of non-closed leaves below α in \mathcal{T} , $\text{Layers}(\mathcal{T}, k)$ denotes the set of layers of rank k

in \mathcal{T} and $\text{Layers}(\mathcal{T}, k, \alpha)$ denotes the set of layers of rank k in \mathcal{T} that occur below α . For any set of formulæ Φ , we denote by $\bigwedge \Phi$ the conjunction $\bigwedge_{\phi \in \Phi} \phi$. If \mathcal{T} is a tableau and N is a set of nodes in \mathcal{T} , then $\mathcal{T}[N]$ denotes the disjunction $\bigvee_{\alpha \in N} \bigwedge \mathcal{T}(\alpha)$. We write $\text{cnf}(\phi)$ for a (subsumption-minimal) clausal form of $\phi \downarrow_{\mathcal{R}}$.

Definition 20 A tableau \mathcal{T} is *globally looping* (w.r.t. two natural numbers k and n) iff the following conditions hold:

1. $n < k$.
2. $\mathcal{T}[\text{Layers}(\mathcal{T}, k)] = \mathcal{T}[\text{Layers}(\mathcal{T}, n)]$ (modulo AC and idempotence).
3. All non-closed leaves in \mathcal{T} are of a rank greater or equal to k .

Then the *Global Loop Detection rule* closes every node in $\text{Layers}(\mathcal{T}, k)$. \diamond

By definition, after the Global Loop Detection rule is applied, all branches containing the parameter n are closed and the construction of the tableau is over (since no leaf can be expanded anymore). Note that the Global Loop Detection rule can be simulated by several applications of the Loop Detection rule introduced in Section 2. Indeed, assume that a pair of natural numbers (k, n) satisfying the conditions of Definition 20 exists. Then, by Condition 2, for every layer α of rank k , there exists a layer β of rank n such that $\mathcal{T}(\alpha) = \mathcal{T}(\beta)$. Thus the Loop Detection rule applies on α (w.l.o.g. we assume that the layers of rank n are constructed before those of rank k in all parallel branches, which is possible since $n < k$). However, it is easy to see that the converse does not hold: the Global Loop Detection rule is *strictly less general* than the looping rule. It is, however, powerful enough to ensure termination, provided that a fair strategy is used to expand the tableau, as stated by the following theorem:

Theorem 21 Let $(\mathcal{T}_i)_{i \in \mathbb{N}}$ be an infinite sequence of tableaux such that, for every $i \in \mathbb{N}$, \mathcal{T}_{i+1} is obtained from \mathcal{T}_i by applying one of the Expansion rules of Section 2, other than the Loop Detection rule. Assume, moreover, that for every $k \in \mathbb{N}$, there exists $n \in \mathbb{N}$ such that every non-closed leaf in \mathcal{T}_n is of a rank greater than k (i.e. no branch is indefinitely “frozen”, the rank of the leaves increases indefinitely). There exists $n \in \mathbb{N}$ such that \mathcal{T}_n is globally looping.

PROOF. We have shown (see the proof of Theorem 5) that the number of sets $\mathcal{T}_i(\alpha)$ where α is a leaf in \mathcal{T}_i is finite. Thus the set of sets of nodes $\{\mathcal{T}_i(\alpha) \mid \alpha \in \text{Layers}(\mathcal{T}_i, n)\}$ is also finite. Let k be a natural number that is strictly greater than the cardinality of this set. By the hypothesis of the theorem, there exists $n \in \mathbb{N}$ such that every leaf in \mathcal{T}_n is of a rank greater than k . By the pigeonhole argument, there exist two natural numbers $n' < k'$ such that $\mathcal{T}[\text{Layers}(\mathcal{T}_n, n')] = \mathcal{T}[\text{Layers}(\mathcal{T}_n, k')]$. Then \mathcal{T}_n is globally looping. \blacksquare

We now show that from every tableau \mathcal{T} , one can extract a resolution derivation from the root of \mathcal{T} of the disjunction of the leaves of \mathcal{T} . We first restrict ourselves to tableaux built without the Explosion and Loop Detection rules. We

focus on such tableaux because they correspond to the subtrees that are found “between” two layers in an tableau built without restriction on the rules. More precisely, take a layer α of some rank m in a tableau \mathcal{T} (built without restriction on the rules). Then the subtree of \mathcal{T} of root α and whose leaves are the layers of rank $m+1$ below α is indeed a tree built without Explosion nor Loop Detection (by definition of a layer).

We first build derivations for such subtrees, those derivations will then be used as the base elements for building the final schema of refutation. For such a tree \mathcal{T} and a node α of \mathcal{T} , the next definition introduces $\Delta(\mathcal{T}, \alpha)$, which is intended to be a derivation of $\text{cnf}(\mathcal{T}[\text{Leaves}(\mathcal{T}, \alpha)])$ from $\text{cnf}(\mathcal{T}(\alpha))$.

Definition 22 Let \mathcal{T} be a tableau constructed using the Expansion rules, *except* the Explosion and Loop Detection rules. Let α be a node in \mathcal{T} . We define a derivation $\Delta(\mathcal{T}, \alpha)$ inductively, according to the rule that is applied on α :

- If α is a leaf, then $\Delta(\mathcal{T}, \alpha)$ is defined as the sequence of clauses in $\text{cnf}(\mathcal{T}(\alpha))$.
- If the Closure rule is applied on α , using two formulæ ϕ and $\neg\phi$, then $\Delta(\mathcal{T}, \alpha) \stackrel{\text{def}}{=} \phi \cdot \neg\phi \cdot \perp$ (notice that since the formulæ are in NNF, ϕ must be an atom).
- If the Normalisation, Purity or \wedge -Decomposition rule is applied on α , yielding a node β then $\Delta(\mathcal{T}, \alpha) \stackrel{\text{def}}{=} \Delta(\mathcal{T}, \beta)$.
- Finally, assume that the \vee -Decomposition rule is applied on α yielding two nodes β_1 and β_2 . Let Φ_1 and Φ_2 be the clausal forms of ϕ_1 and ϕ_2 respectively. For any $C \in \Phi_2$, let $\Lambda'(C)$ be the derivation obtained from $\Delta(\mathcal{T}, \beta_1)$ by replacing every occurrence of a clause $D \in \Phi_1$ by $D \vee C$ (and by adding the disjunction $\vee C$ to every descendant of D).

For any clause C' in $\text{cnf}(\mathcal{T}[\text{Leaves}(\mathcal{T}, \beta_1)])$, we construct a derivation $\Lambda''(C')$ from $\Delta(\mathcal{T}, \beta_2)$ by replacing every occurrence of a clause $D \in \Phi_2$ by $D \vee C'$ (and by adding the disjunction $\vee C'$ to every descendant of D). Then $\Delta(\mathcal{T}, \alpha)$ is the concatenation of all the derivations $\Lambda'(C)$ and $\Lambda''(C')$ (with $C \in \Phi_2$ and $C' \in \text{cnf}(\mathcal{T}[\text{Leaves}(\mathcal{T}, \beta_1)])$).

Only the case of disjunction is non-trivial. Informally, it does nothing more than building, for two sets of clauses S_1 and S_2 , a derivation of $\text{cnf}(S_1 \vee S_2)$ from two derivations of S_1 and S_2 .

The following lemma states that $\Delta(\mathcal{T}, \alpha)$ satisfies the desired property:

Lemma 23 *Let \mathcal{T} be a tableau, constructed by using the previous expansion rules, except the Explosion and Loop Detection rules. For all nodes α in \mathcal{T} , $\Delta(\mathcal{T}, \alpha)$ is a derivation of $\text{cnf}(\mathcal{T}[\text{Leaves}(\mathcal{T}, \alpha)])$ from $\text{cnf}(\mathcal{T}(\alpha))$.*

PROOF. The proof is by induction on the depth of \mathcal{T} . We distinguish several cases, according to the rule applied on α .

- If α is a leaf then $\text{Leaves}(\mathcal{T}, \alpha) = \{\alpha\}$. Moreover, according to Definition 22, $\Delta(\mathcal{T}, \alpha)$ is the sequence of formulæ in $\text{cnf}(\mathcal{T}(\alpha))$, thus the proof is completed.
- If the Normalisation or \wedge -Decomposition rule is applied on α , yielding a node β , then we have $\text{cnf}(\mathcal{T}(\alpha)) = \text{cnf}(\mathcal{T}(\beta))$. Moreover, since α has only one child, $\text{cnf}(\mathcal{T}[\text{Leaves}(\mathcal{T}, \alpha)]) = \text{cnf}(\mathcal{T}[\text{Leaves}(\mathcal{T}, \beta)])$. Hence the proof is immediate, by the induction hypothesis.
- If the Purity rule is applied on α , using a formula ϕ , yielding a node β , then by the induction hypothesis, $\Delta(\mathcal{T}, \beta)$ is a derivation of $\text{cnf}(\mathcal{T}[\text{Leaves}(\mathcal{T}, \beta)])$ from $\text{cnf}(\mathcal{T}(\beta))$. Since α has only one child, $\text{cnf}(\mathcal{T}[\text{Leaves}(\mathcal{T}, \alpha)]) = \text{cnf}(\mathcal{T}[\text{Leaves}(\mathcal{T}, \beta)])$. Furthermore, $\text{cnf}(\mathcal{T}(\alpha))$ is of the form $\phi \wedge \text{cnf}(\mathcal{T}(\beta))$ thus $\Delta(\mathcal{T}, \beta)$ is also a derivation from $\text{cnf}(\mathcal{T}(\alpha))$. Since, by Definition 22, $\Delta(\mathcal{T}, \alpha) = \Delta(\mathcal{T}, \beta)$, the proof is completed.
- Finally, assume that the Disjunction rule is applied on α , using a formula $\phi_1 \vee \phi_2$. This yields two nodes β_1 and β_2 , corresponding respectively to ϕ_1 and ϕ_2 . Let Φ_1 and Φ_2 be a cnf of ϕ_1 and ϕ_2 respectively. By definition, $\text{cnf}(\mathcal{T}[\text{Leaves}(\mathcal{T}, \alpha)])$ is the clausal form of the disjunction of $\text{cnf}(\mathcal{T}[\text{Leaves}(\mathcal{T}, \beta_1)])$ and $\text{cnf}(\mathcal{T}[\text{Leaves}(\mathcal{T}, \beta_2)])$, hence every clause occurring in $\text{cnf}(\mathcal{T}[\text{Leaves}(\mathcal{T}, \alpha)])$ is of the form $C_1 \vee C_2$ where C_i occurs in $\text{cnf}(\mathcal{T}[\text{Leaves}(\mathcal{T}, \beta_i)])$ ($i = 1, 2$). By the induction hypothesis $\Delta(\mathcal{T}, \beta_1)$ is a derivation of $\text{cnf}(\mathcal{T}[\text{Leaves}(\mathcal{T}, \beta_1)])$ from $\text{cnf}(\mathcal{T}(\beta_1))$. Thus in particular, for every $C \in \Phi_2$, $\Lambda'(C)$ (see Definition 22 for the notations) is a derivation from $\text{cnf}(\mathcal{T}(\alpha))$ of either C_1 or $C_1 \vee C$. In the first case, the formula ϕ_1 is not needed for deriving C_1 , thus actually, C_1 also occurs in $\text{cnf}(\mathcal{T}[\text{Leaves}(\mathcal{T}, \beta_2)])$. Since $C_1 \vee C_2$ is subsumption-minimal, we must have actually $C_1 = C_2$ and the proof is completed. In the second case, by the induction hypothesis $\Delta(\mathcal{T}, \beta_2)$ is a derivation of C_2 from $\text{cnf}(\mathcal{T}(\beta_2))$, thus $\Lambda''(C_1)$ is a derivation of $C_1 \vee C_2$ from $\text{cnf}(\mathcal{T}(\alpha)) \cup \{C_1 \vee D \mid D \in \Phi_2\}$. Hence $\Delta(\mathcal{T}, \alpha)$ is a derivation of $C_1 \vee C_2$ from $\text{cnf}(\mathcal{T}(\alpha))$.

■

Thus the function $\mathcal{T}(\alpha) \rightarrow \Delta(\mathcal{T}, \alpha)$ allows us to build derivations from subtrees of a whole tableau. Intuitively, the next step is to put together those derivations according to the positions of the corresponding subtrees in the main tableau. Consider a rank m in a tableau \mathcal{T} . One can apply the function Δ to all the (parallel) subtrees whose root is a layer of rank m . Then we can do the same at rank $m + 1$, append every resulting derivation to the derivation obtained from the parent tree, and go on at rank $m + 2$, etc. This intuitively gives the structure of a rewrite system where n decreases each time we go to the next rank. However this gives us a tree-like structure (to every derivation corresponding to a subtree \mathcal{U} we append the derivations corresponding to all the leaves of \mathcal{U} , and go on with the trees below those leaves) similar to the rewrite systems presented in Section 3. Instead we would like a more linear structure. So we will consider *at once* all the layers of a given rank and get only

one derivation corresponding to those nodes. For this, we need a way to apply Δ to all the subtrees at once. This is actually done by building a new tableau from the subtrees.

Let \mathcal{T} be a tableau of root α . Assume that \mathcal{T} is globally looping w.r.t. n and k , with $n < k$. Let $m < k$. We denote by $\mathcal{U}(\mathcal{T}, m)$ a tableau whose root is labeled by a formula $\mathcal{T}[\text{Layers}(\mathcal{T}, m)]$ (note that we take *all the layers* of rank m at a time), and obtained by applying the \vee and \wedge -Decomposition and Closure rules (and only these rules) until irreducibility. By definition, since the root formula of $\mathcal{U}(\mathcal{T}, m)$ is the disjunction of the labels of the layers in $\text{Layers}(\mathcal{T}, m)$, every non-closed leaf β of $\mathcal{U}(\mathcal{T}, m)$ is labeled by a set of formulæ of the form $\mathcal{T}(\gamma_\beta)$, where $\gamma_\beta \in \text{Layers}(\mathcal{T}, m)$. Furthermore, for every $\gamma \in \text{Layers}(\mathcal{T}, m)$, there exists a leaf β of $\mathcal{U}(\mathcal{T}, m)$ such that $\gamma_\beta = \gamma$. Since $m < k$ and since by Definition 20 the leaves of \mathcal{T} must be of a rank greater or equal to k , the node γ_β cannot be a leaf of \mathcal{T} . This implies that some rule is applied on γ_β . But the only rule that is applicable on a layer (beside the Global Loop Detection rule that cannot be applied on layers of a rank distinct from k) is the Explosion rule. Hence \mathcal{T} necessarily contains two subtableaux, written \mathcal{T}_β^0 and \mathcal{T}_β^1 , of roots $\mathcal{T}(\gamma_\beta)\{\mathbf{n} \leftarrow 0\}$ and $\mathcal{T}(\gamma_\beta)\{\mathbf{n} \leftarrow \mathbf{n} + 1\}$ respectively. Then $\mathcal{V}^0(\mathcal{T}, m)$ and $\mathcal{V}^1(\mathcal{T}, m)$ denote respectively the tableaux obtained from $\mathcal{U}(\mathcal{T}, m)\{\mathbf{n} \leftarrow 0\}$ and $\mathcal{U}(\mathcal{T}, m)\{\mathbf{n} \leftarrow \mathbf{n} + 1\}$ by:

- Replacing every leaf β by \mathcal{T}_β^0 and \mathcal{T}_β^1 respectively.
- Removing, in the obtained tableau, all applications of the Explosion rule³ (and all the nodes that occur below such an application).

By definition, the leaves of $\mathcal{V}^0(\mathcal{T}, m)$ and $\mathcal{V}^1(\mathcal{T}, m)$ are layers. They correspond either to the leaves of \mathcal{T} or to the nodes in \mathcal{T} on which Explosion is applied (these nodes are of rank $m + 1$ in \mathcal{T}).

Proposition 24 *Let \mathcal{T} be a tableau that is globally looping w.r.t. two numbers $n < k$. Let $m < k$. For any non closed leaf β of $\mathcal{U}(\mathcal{T}, m)$, \mathcal{T}_β^0 is closed and $\text{Layers}(\mathcal{T}_\beta^1, 0) = \text{Layers}(\mathcal{T}, m + 1, \gamma_\beta)$.*

PROOF. By definition, all leaves not containing \mathbf{n} in \mathcal{T} must be closed. Thus \mathcal{T}_β^0 is closed. Furthermore, by definition, the layers of rank 0 in \mathcal{T}_β^1 are the first layers of every branch, i.e. the first layer after γ_β in \mathcal{T} . Since γ_β is a layer of rank m in \mathcal{T} , such layers are of rank $m + 1$. ■

Corollary 25 *Let \mathcal{T} be a tableau that is globally looping w.r.t. two numbers $n < k$. Let $m < k$. Let β and β' be the roots of $\mathcal{V}^0(\mathcal{T}, m)$ and $\mathcal{V}^1(\mathcal{T}, m)$ respectively. $\text{cnf}(\mathcal{T}[\text{Leaves}(\mathcal{V}^0(\mathcal{T}, m), \beta)]) = \perp$ and $\text{cnf}(\mathcal{T}[\text{Leaves}(\mathcal{V}^1(\mathcal{T}, m), \beta)]) = \text{cnf}(\mathcal{T}[\text{Layers}(\mathcal{T}, m + 1)])$.*

³Note that, although no application of the Explosion rule occurs in $\mathcal{U}(\mathcal{T}, m)$, some applications of this rule may occur in \mathcal{T}_β^1 .

PROOF. The first point stems directly from Proposition 24. For the second point, we only have to remark that by definition a node occurs in $\text{Layers}(\mathcal{T}, m+1)$ iff it occurs in some set $\text{Layers}(\mathcal{T}, m+1, \gamma_\beta)$, where β is a leaf of $\mathcal{U}(\mathcal{T}, m)$ (since the leaves of $\mathcal{U}(\mathcal{T}, m)$ are exactly the layers of rank m in \mathcal{T}). ■

By applying the above function $\Delta(\mathcal{T}, \alpha)$ on the two tableaux $\mathcal{V}^1(\mathcal{T}, m)$ and $\mathcal{V}^0(\mathcal{T}, m)$, we define the following derivations (where α denotes the root of $\mathcal{V}^1(\mathcal{T}, m)$ and $\mathcal{V}^0(\mathcal{T}, m)$):

$$\Lambda^1(\mathcal{T}, m) \stackrel{\text{def}}{=} \Delta(\mathcal{V}^1(\mathcal{T}, m), \alpha) \quad \Lambda^0(\mathcal{T}, m) \stackrel{\text{def}}{=} \Delta(\mathcal{V}^0(\mathcal{T}, m), \alpha)$$

The following lemma states essential properties of $\Lambda^1(\mathcal{T}, m)$ and $\Lambda^0(\mathcal{T}, m)$:

Lemma 26 *Let \mathcal{T} be a tableau that is globally looping w.r.t. two numbers $n < k$. Let $m < k$.*

- $\Lambda^0(\mathcal{T}, m)$ is a refutation of $\text{cnf}(\mathcal{T}[\text{Layers}(\mathcal{T}, m)])\{\mathbf{n} \leftarrow 0\}$.
- If $m < k-1$ then $\Lambda^1(\mathcal{T}, m)$ is a derivation from $\text{cnf}(\mathcal{T}[\text{Layers}(\mathcal{T}, m)])\{\mathbf{n} \leftarrow \mathbf{n} + 1\}$ of $\text{cnf}(\mathcal{T}[\text{Layers}(\mathcal{T}, m+1)])$.
- $\Lambda^1(\mathcal{T}, k-1)$ is a derivation from $\text{cnf}(\mathcal{T}[\text{Layers}(\mathcal{T}, m)])\{\mathbf{n} \leftarrow \mathbf{n} + 1\}$ of $\text{cnf}(\mathcal{T}[\text{Layers}(\mathcal{T}, n)])$.

PROOF. Let β and β' be the roots of $\mathcal{V}^1(\mathcal{T}, k)$ and $\mathcal{V}^0(\mathcal{T}, k)$ respectively. By Lemma 23, $\Lambda^1(\mathcal{T}, m)$ is a derivation from $\text{cnf}(\mathcal{V}^1(\mathcal{T}, m)(\beta))$ of $\text{cnf}(\mathcal{T}[\text{Leaves}(\mathcal{V}^1(\mathcal{T}, m), \beta)])$. By definition of $\mathcal{V}^1(\mathcal{T}, m)$, the root of $\mathcal{V}^1(\mathcal{T}, m)$ is labeled by $\Phi\{\mathbf{n} \leftarrow \mathbf{n} + 1\}$, where Φ is the root of $\mathcal{U}(\mathcal{T}, m)$. By definition of $\mathcal{U}(\mathcal{T}, m)$, $\Phi = \mathcal{T}[\text{Layers}(\mathcal{T}, m)]$. Hence $\Lambda^1(\mathcal{T}, k)$ is a derivation from $\text{cnf}(\mathcal{T}[\text{Layers}(\mathcal{T}, m)])\{\mathbf{n} \leftarrow \mathbf{n} + 1\}$. Similarly, $\Lambda^0(\mathcal{T}, m)$ is a derivation from $\text{cnf}(\mathcal{T}[\text{Layers}(\mathcal{T}, m)])\{\mathbf{n} \leftarrow 0\}$.

By Corollary 25, $\text{cnf}(\mathcal{T}[\text{Leaves}(\mathcal{V}^1(\mathcal{T}, m), \beta)]) = \text{cnf}(\mathcal{T}[\text{Layers}(\mathcal{T}, m+1)])$. Furthermore, if $m = k-1$, then since \mathcal{T} is globally looping we have $\text{cnf}(\mathcal{T}[\text{Layers}(\mathcal{T}, m+1)]) = \text{cnf}(\mathcal{T}[\text{Layers}(\mathcal{T}, n)])$.

Similarly, $\text{cnf}(\mathcal{T}[\text{Leaves}(\mathcal{V}^1(\mathcal{T}, m), \beta)]) = \perp$. ■

Let \mathcal{T} be a tableau that is globally looping w.r.t. two numbers $n < k$. We associate to each natural number $m < k$ a symbol γ^m . Let $\mathfrak{R}^*(\mathcal{T})$ the system containing the following rules. Note that $\mathcal{V}^0(\mathcal{T}, m)$ and $\mathcal{V}^1(\mathcal{T}, m)$ are defined only w.r.t. the rank m , but not w.r.t. a particular node. Thus, contrarily to the transformation of Section 3, there is not one derivation per node, but rather one derivation per rank.

$$\gamma_0^m \rightarrow \Lambda^0(\mathcal{T}, m) \quad \gamma_{\mathbf{n}+1}^m \rightarrow \Lambda^1(\mathcal{T}, m) \cdot \gamma_{\mathbf{n}}^{m+1} \text{ (if } m+1 < k) \quad \gamma_{\mathbf{n}+1}^{k-1} \rightarrow \Lambda^1(\mathcal{T}, k) \cdot \gamma_{\mathbf{n}}^n$$

Intuitively, we are appending the derivations, rank after rank, until we reach the rank k where the Global Loop Detection applies. In this case we get back at

the rank of looping n . Thus we can see the use of grouping the derivations by rank (instead of node) as it allows to benefit from the simplified form of looping induced by the Global Loop Detection rule. In the end, the resulting rewrite system is indeed much simpler.

Proposition 27 $\mathfrak{R}^*(\mathcal{T})$ is convergent.

PROOF. Termination is easy to obtain since the rules in $\mathfrak{R}^*(\mathcal{T})$ strictly decreases the value of the indices of the symbols γ^k . Furthermore, $\mathfrak{R}^*(\mathcal{T})$ is obviously orthogonal. ■

Note that, by definition, $\mathfrak{R}^*(\mathcal{T})$ is always propositional (unlike $\mathfrak{R}(\mathcal{T})$).

Theorem 28 Let \mathcal{T} be a tableau of root α that is globally looping w.r.t. two numbers n, k , with $n < k$. Let $m < k$. For all $i \in \mathbb{N}$, $\gamma_i^m \downarrow_{\mathfrak{R}^*(\mathcal{T})}$ is a refutation of $\text{cnf}(\mathcal{T}[\text{Layers}(\mathcal{T}, m)])\{\mathbf{n} \leftarrow i\} \downarrow_{\mathcal{R}}$. Thus in particular, if α is a layer, $\gamma_i^0 \downarrow_{\mathfrak{R}^*(\mathcal{T})}$ is a refutation of $\mathcal{T}(\alpha)\{\mathbf{n} \leftarrow i\} \downarrow_{\mathcal{R}}$.

PROOF. This follows by induction on i . If $i = 0$ then we have, by definition of the rules in $\mathfrak{R}^*(\mathcal{T})$: $\gamma_i^m \downarrow_{\mathfrak{R}^*(\mathcal{T})} = \Lambda^0(\mathcal{T}, m) \downarrow_{\mathcal{R}}$. By Lemma 26 (first point), $\Lambda^0(\mathcal{T}, m) \downarrow_{\mathcal{R}}$ is a refutation of $\text{cnf}(\mathcal{T}[\text{Layers}(\mathcal{T}, m)])\{\mathbf{n} \leftarrow 0\} \downarrow_{\mathcal{R}}$.

If $i > 0$ then we have $\gamma_i^m \downarrow_{\mathfrak{R}^*(\mathcal{T})} = \Lambda^1(\mathcal{T}, m) \downarrow_{\mathcal{R}} \{\mathbf{n} \leftarrow i\} \cdot \gamma_{i-1}^{m+1} \downarrow_{\mathfrak{R}^*(\mathcal{T})}$. If $m < k - 1$, then by Lemma 26 (second point), $\Lambda^1(\mathcal{T}, m)$ is a derivation from $\text{cnf}(\mathcal{T}[\text{Layers}(\mathcal{T}, m)])\{\mathbf{n} \leftarrow \mathbf{n} + 1\}$ of $\text{cnf}(\mathcal{T}[\text{Layers}(\mathcal{T}, m + 1)])$, hence $\Lambda^1(\mathcal{T}, m) \downarrow_{\mathcal{R}}$ is a derivation from $\text{cnf}(\mathcal{T}[\text{Layers}(\mathcal{T}, m)])\{\mathbf{n} \leftarrow i\} \downarrow_{\mathcal{R}}$ of $\text{cnf}(\mathcal{T}[\text{Layers}(\mathcal{T}, m + 1)])\{\mathbf{n} \leftarrow i - 1\} \downarrow_{\mathcal{R}}$. Then by the induction hypothesis, $\gamma_{i-1}^{m+1} \downarrow_{\mathfrak{R}^*(\mathcal{T})}$ is a refutation of $\text{cnf}(\mathcal{T}[\text{Layers}(\mathcal{T}, m + 1)])\alpha\{\mathbf{n} \leftarrow i - 1\} \downarrow_{\mathcal{R}}$.

If $m = k - 1$, then by Lemma 26 (second point), $\Lambda^1(\mathcal{T}, m)$ is a derivation from $\text{cnf}(\mathcal{T}[\text{Layers}(\mathcal{T}, m)])\{\mathbf{n} \leftarrow \mathbf{n} + 1\}$ of $\text{cnf}(\mathcal{T}[\text{Layers}(\mathcal{T}, n)])$, hence $\Lambda^1(\mathcal{T}, m) \downarrow_{\mathcal{R}}$ is a derivation from $\text{cnf}(\mathcal{T}[\text{Layers}(\mathcal{T}, m)])\{\mathbf{n} \leftarrow i\} \downarrow_{\mathcal{R}}$ of $\text{cnf}(\mathcal{T}[\text{Layers}(\mathcal{T}, n)])\{\mathbf{n} \leftarrow i - 1\} \downarrow_{\mathcal{R}}$. Then by the induction hypothesis, $\gamma_{i-1}^n \downarrow_{\mathfrak{R}^*(\mathcal{T})}$ is a refutation of $\text{cnf}(\mathcal{T}[\text{Layers}(\mathcal{T}, n)])\alpha\{\mathbf{n} \leftarrow i - 1\} \downarrow_{\mathcal{R}}$.

When α is not a layer, the rewrite system is easily adapted by prepending the derivation obtained by applying Δ to the subtree of \mathcal{T} whose leaves are the layers of rank 0.

Example 29 Consider the tableau of Example 19. This tableau is actually globally looping. The following rewrite system is constructed (after partial evaluation and simplification):

$$\begin{aligned} \gamma_0 &\rightarrow p_0 \vee q_0 \cdot \neg p_0 \cdot q_0 \cdot \neg q_0 \cdot \perp \\ \gamma_{n+1} &\rightarrow (p_{n+1} \vee q_{n+1}) \cdot (q_n \vee \neg p_{n+1}) \cdot (q_n \vee q_{n+1}) \cdot (p_n \vee \neg q_{n+1}) \cdot (q_n \vee p_n) \cdot \gamma_n \end{aligned}$$

Compared with the system produced by the previous method (see Example 19), these rules are obviously simpler (no schema variable are needed, and only linear recursion is used). Furthermore, it is easy to check that they generate much shorter derivations. ♣

5 Conclusion

Two distinct algorithms have been designed for extracting schemata of resolution proofs from closed tableaux. This work is motivated by the fact that such refutations are needed for some natural applications of schemata calculus (unsatisfiability detection is not always sufficient). In particular, the explicit generation of the proofs (even in the form of proof schemata) makes possible the certification of the results produced by the provers. The first algorithm tackles the tableau calculus in its full generality, but it yields very complex representations of the derivations (which will make them less usable in practice, in particular they are not very informative for a human user). The second one uses a less powerful calculus, but it generates schemata of refutations in a much simpler format (propositional rewrite systems are obtained).

There is thus a natural trade-off between the two presented methods: none of them is uniformly superior to the other. The choice between the two algorithms should be made according to the considered applications, and/or to the form of the constructed tableaux. In some cases, as shown by the examples in Section 3, the first approach generates a propositional rewrite system. In this case it should of course be preferred. Future work includes the implementation of the two methods and the precise evaluation of the complexity of the second algorithm. One could also wonder whether a polynomial algorithm generating propositional derivations exists for the general case. We conjecture that the use of Δ -variables cannot be avoided in general.

References

- [1] ASAP: About schemata and proofs. ANR-FWF Research project, 2010–2013. <http://membres-lig.imag.fr/peltier/ASAP/>.
- [2] V. Aravantinos, R. Caferra, and N. Peltier. A schemata calculus for propositional logic. In *TABLEAUX 09 (International Conference on Automated Reasoning with Analytic Tableaux and Related Methods)*, volume 5607 of *LNCS*, pages 32–46. Springer, 2009.
- [3] V. Aravantinos, R. Caferra, and N. Peltier. A Decidable Class of Nested Iterated Schemata. In *IJCAR 2010 (International Joint Conference on Automated Reasoning)*, *LNCS*, pages 293–308. Springer, 2010.
- [4] V. Aravantinos, R. Caferra, and N. Peltier. Decidability and undecidability results for propositional schemata. *Journal of Artificial Intelligence Research*, 40:599–656, 2011.
- [5] M. Baaz and A. Leitsch. Cut-elimination and Redundancy-elimination by Resolution. *Journal of Symbolic Computation*, 29(2):149–176, 2000.
- [6] O. Bar-Ilan, O. Fuhrmann, S. Hoory, O. Shacham, and O. Strichman. Linear-time reductions of resolution proofs. In *Proceedings of HVC’08*.
- [7] P. Beame, H. Kautz, and A. Sabharwal. Towards understanding and harnessing the potential of clause learning. *Journal of Artificial Intelligence Research*, 22:319–351, 2004.

- [8] A. Cimatti, A. Griggio, and R. Sebastiani. A simple and flexible way of computing small unsatisfiable cores in SAT modulo theories. In *SAT*, LNCS, pages 334–339. Springer, 2007.
- [9] A. Leitsch. *The resolution calculus*. Springer. Texts in Theoretical Computer Science, 1997.
- [10] K. L. McMillan and N. Amla. Automatic abstraction without counterexamples. In *Proceedings of TACAS'03*, pages 2–17. Springer-Verlag, 2003.
- [11] G. Mints. Proof theory in the USSR 1925-1969. *J. Symb. Log.*, 56(2):385–424, 1991.
- [12] R. M. Smullyan. *First-Order Logic*. Springer, 1968.
- [13] A. Wolf. Optimization and translation of tableau-proofs into resolution. *Journal of Information Processing and Cybernetics*, 30(5/6):311–325, 1994.
- [14] L. Zhang and S. Malik. Validating SAT Solvers Using an Independent Resolution-Based Checker: Practical Implementations and Other Applications. In *DATE*, pages 10880–10885. IEEE Computer Society, 2003.