A syntactic soundness proof for free-variable tableaux with on-the-fly Skolemization

Richard Bonichon[1] and Olivier Hermant[2]

[1] Universidade Federal do Rio Grande do Norte, Natal, RN, Brazil
richard@dimap.ufrn.br
[2] MINES ParisTech, PSL Research University, France
olivier.hermant@mines-paristech.fr

Abstract. We prove the syntactic soundness of classical tableaux with free variables and on-the-fly Skolemization. Soundness proofs are usually built from semantic arguments, and this is to our knowledge, the first proof that appeals to syntactic means. We actually prove the soundness property with respect to cut-free sequent calculus. This requires great care because of the additional liberty in freshness checking allowed by the use of Skolem terms. In contrast to semantic soundness, we gain the possibility to state a cut elimination theorem for sequent calculus, under the proviso that completeness of the method holds. We believe that such techniques can be applied to tableaux in other logics as well.

1 Introduction

Tableaux methods form a successful sub-family of automated theorem proving, encompassing classical as well as modal logics. Their origin comes from Beth's semantic considerations [3]. With Smullyan's updated tree-based formalism [16], as well as Fitting's subsequent treatment [11], there is a first separation between syntactic and semantic concerns. Both present a purely syntactic operational behavior of tableaux rules, justified by semantic soundness and completeness proofs. Proving these two properties by semantic arguments has stayed the norm and for good reasons: model-theoretic proofs are reasonably short, relatively elegant and straightforward. In comparison, syntactic proofs can be messy, as all translation details must be shown.

There might be another reason. Translating ground tableaux proofs à la Smullyan to ground sequent calculus proofs is indeed trivial. If we allow free variables and Skolemization, we still have a straightforward translation to Antonsen and Waaler's free-variable sequent calculus [17]. Thus, the relation between classical tableaux and sequent calculi has been relegated to folklore knowledge.

Nonetheless, translating free variable tableaux with Skolemization to ground sequent calculus is not as simple a task: most of the trouble comes from the freshness conditions imposed on existential witnesses in sequents. Despite our efforts, we were not able to find any result on that matter.

However, why would one want syntactic soundness over semantic soundness ? At the proof-theoretical level, it provides a double-check of soundness. In practice, it does not add any power to tableaux heuristics. However, it presents some benefits, especially in the context of proof production and proof theory.

Since it is not hard to encode ground sequent calculus rules into any proof assistant such as Coq, Isabelle or Dedukti [4], if we are able to reconstruct a ground sequent derivation from a free-variable Skolemized tableaux procedure, we will get (almost) free external verification tools. On the tableaux side, a syntactic soundness proof highlights where and how non-elementary speedups are achieved from the use of efficient δ-rules. Lastly, our long-term goal is to derive cut elimination theorems from tableaux completeness proofs, in extensions of first-order logic, and this requires syntactic, cut-free, soundness proofs.

2 Free-Variable Tableaux

The language is usual first-order logic with predicate and function symbols. Sets and multisets of formulas are denoted by capital greek letters (Γ, Δ), while formulas are denoted by upper case letters A, B, C, D. We use the lower case letters f, g to denote function symbols and a, b, c, d for constants. Variables are denoted as x, y, z. We also use indexes or quotes when we need more symbols.

We present tableaux as a refutation calculus with attached constraints via a global constraint store. This global store represents the necessary unification steps to be performed and satisfied in order to close the tableau. A *constrained tableau* is a pair $\mathcal{T} \cdot \mathcal{C}$ where $\mathcal{T}$ is a tableau and $\mathcal{C}$ a set of unification constraints.

A branch can be *closed* when it carries two opposite unifiable formulas. *Unifiable* here means that the global store does not become inconsistent when adding the new unification constraints. A tableau is itself said closed when *all its branches* can be closed at once. In this case, all closing constraints are unifiable.

This means that closing a first-order tableau can be seen as providing a unifier that simultaneously satisfies all the global constraints *and* the closing constraints of the open branches, or, equivalently, that does not induce any new constraint on the latter branches. The constraint store keeps the minimal requirements for such a unifier: they come from the early closure of some branches, discussed before. Of course, if this is done carelessly, we can come to a dead-end.

We see constraints as a degree of liberty for tableaux. Ultimately, we just can decide not to generate constraints at all, until a global unifier can be found. The soundness proof of Section 4 promotes this point of view: it assumes a unifier and no constraints.

The rules, presented in Figure 1 where the constraints are omitted if they are unchanged, are an extension of usual non-destructive free-variable tableaux calculi. Non-destructivity is not strictly needed neither for soundness nor for completeness, but it eases some developments.

Tableaux rules are usually divided into 4 sets: 2 sets decompose logical connectives (α, β), two act on quantifiers (δ, γ). We need only add the closure rule ($\odot$). If $\wedge, \vee, \Rightarrow, \neg, \forall, \exists$ are allowed, we have the following groups:

α	$A \wedge B, \neg(A \vee B), \neg(A \Rightarrow B), \neg\neg A$
β	$A \vee B, A \Rightarrow B, \neg(A \wedge B)$
δ	$\exists x\ A, \neg(\forall x\ A)$
γ	$\forall x\ A, \neg(\exists x\ A)$

The decomposition of formulas happens as follows: the tableaux method matches the active formula with one of the above categories, then applies the corresponding rule to it. Negated formulas are actually handled in two steps: the negation is pushed to the direct subformulas, transforming the active connective by De Morgan laws, then the decomposition of the connective is applied.

In pure automated deduction mode, it is enough to keep only the current set of open branches, since the rules apply only on them. This is no more the case if we are interested in exporting the proof in other formats [5]. Moreover, keeping track of previous steps can help us during proof search.

For proof-theoretic purposes, it is convenient to record all the steps of the proof and to consider a tableau derivation as a *tree* rooted at the original multiset of formulas; tableau branches are *nodes*, internal if they already have been applied some rule and external (*leaves*) otherwise; the leaves that are not closed, are *open*, and they constitute the tableau properly speaking. Tableau rules primarily operate on those leaves, extending one of them at a time: rules are recorded as labels of inner nodes. Trees themselves enjoy a notion of branch, that we replace, to prevent confusion, with tableaux branches, by the word *path*.

Due to the non-destructive nature of the rules, the formulas on a path are collected at the leaves. Paths, as well as leaves/branches, will be identified as usual with trees, with sequences of 0 and 1. $b.0$ is the left child of a path b, (or the unique child if there is no branching), and $b.1$ is its right child.

$$\frac{\alpha(A,B)}{A,B}\ \alpha \qquad \frac{\beta(A,B)}{A \mid B}\ \beta \qquad \frac{\gamma(x,A)}{A(x := X)}\ \gamma \quad X \text{ fresh free variable} \qquad \frac{\delta(x,A)}{A[\mathrm{x}{:=}\texttt{sko}(\mathrm{args})]}\ \delta$$

$$\frac{A, \neg A \cdot \mathcal{C}}{\odot \cdot (\mathcal{C} \cup \{A \approx_{\mathcal{E}} A\})}\ \texttt{closure}\ (\odot)$$

Constraints ($\mathcal{C}$) are omitted in α, β, γ, δ.

Fig. 1: Tableau expansion and closure rules.

α-rules and β-rules correspond to the standard ones as found in Smullyan's textbook [16]. They all include negated formulas, as $\neg$ is a primitive connective, and not an operator transforming formulas into negation normal forms.

Free variables are used in γ-rules as placeholders waiting some satisfying term instantiation, usually given by closure. This has a direct effect on the treatment of existential quantifiers as we now must use Skolemization to get a suitable sound witness.

The δ-rule shown is generic and produces a fresh Skolem symbol on-the-fly. This function symbol, here named `sko`, receives the free variables in A as argu-

ments (args). The term is therefore guaranteed to be fresh. We use a standard *inner Skolemization* [14]: the arguments of the Skolem symbol are the free variables actually occurring in the Skolemized formula A. Inner Skolemization is more efficient than outer Skolemization in the sense that it uses only relevant (i.e. fewer) elements as arguments. Such on-the-fly Skolemization can also be replaced by a pre-inner-Skolemization of formulas (this is the δ^{++} rule of [2]), which would be even more efficient on some problems. We chose not to do so because we intend to extend this work to Deduction modulo [9], which does not behaves well with pre-Skolemization, unless we switch to polarized Deduction modulo [8].

Finally, we also have chosen inner Skolemization over other forms of strong quantifier treatments [6, 12][7] because it adds less noise (through technical difficulties) to the syntactic soundness proof of Section 4.

All in all, inner Skolemization is a good tradeoff between efficiency and simplicity. It allows us to expose the techniques that allow us to show syntactic soundness, with the right degree of difficulty.

Let us prove Smullyan's drinker problem, $\exists x(D(x) \Rightarrow \forall y D(y))$, where D is a unary predicate. As usual with tableaux, we actually refute the negation $\neg(\exists x(D(x) \Rightarrow \forall y D(y)))$. The full derivation is shown in Figure 2.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\neg(\exists x(D(x) \Rightarrow \forall y D(y)))}{\neg(\exists x(D(x) \Rightarrow \forall y D(y))), \neg(D(X) \Rightarrow \forall y D(y))}\,\gamma}{\neg(\exists x(D(x) \Rightarrow \forall y D(y))), \neg(D(X) \Rightarrow \forall y D(y)), D(X), \neg\forall y D(y)}\,\alpha}{\neg(\exists x(D(x) \Rightarrow \forall y D(y))), \neg(D(X) \Rightarrow \forall y D(y)), D(X), \neg\forall y D(y), \neg D(c)}\,\delta}{\odot \quad \{X \approx c\}}\,\odot$$

Fig. 2: A proof of the drinker principle

3 Sequent Calculus

This section presents the sequent calculus which will be used for the syntactic soundness proof for tableaux. This version is as close as possible to tableaux and equivalent to more usual sequent calculi. The important difference with tableaux is that, as most sequent calculi[3], we do not allow free variables nor Skolemization, which will be the major concern of Section 4).

GS3[4] (for Gentzen-Schütte) is a one-sided variant of Gentzen's original LK sequent calculus. Contraction is implicit, built into each inference rule, both to stick to tableaux rules, and as a convenience for the proofs we will develop. In contrast, the weakening rule is explicit. The cut rule is absent, as we intend to go

[3] one exception is Waaler and Antonsen's free-variable sequent calculus[17]
[4] We follow Troelstra and Schwichtenberg's classification and naming [15]

without it in the soundness proof. To underline the similarities with tableaux, we split the presentation of the rules along the $\alpha, \beta, \gamma, \delta$ (Figure 3) classification for tableaux, except that we explicitly mention every case, which is more customary in sequent calculi.

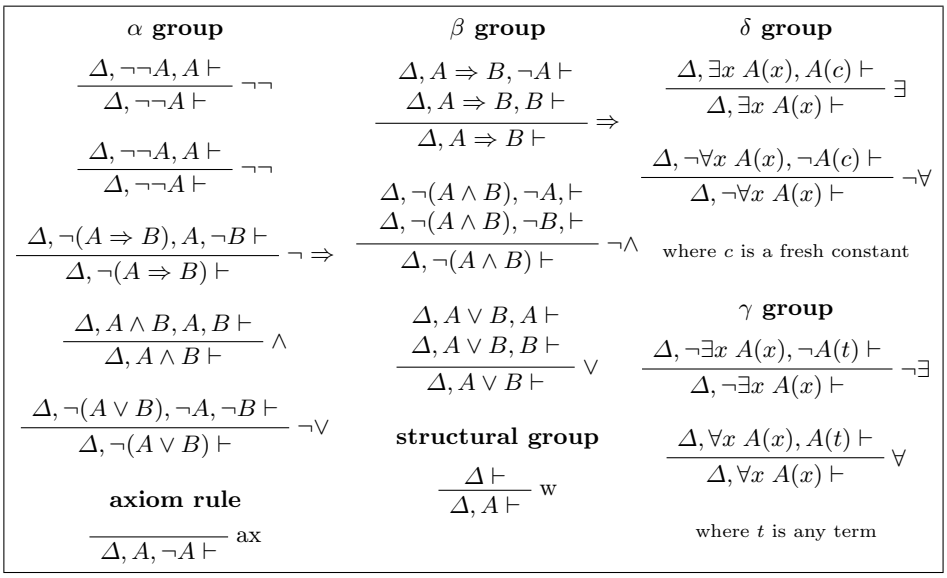

Fig. 3: GS3

4 Soundness Proof

This section shows the following property:

Theorem 1 (Soundness of tableaux w.r.t. GS3). *Let Γ be a set of formulas. If there is a closed tableau rooted at Γ, with unifier σ, then the sequent $\sigma\Gamma \vdash$ has a GS3 proof.*

We require a closed tableau proof, that is to say an entire tree (see Section 2) where all branches are closed and the constraints from the last generated constraint store (the last rule is closure) are satisfiable at once by some unifier σ. It also satisfies any intermediate constraint from this tableau proof as they all appear in the final store.

The unifier σ can assign any term, including a free variable, to a given free variable. To make it ground, we extend it to $\sigma' = \kappa \circ \sigma$, where κ maps the free variables from the range of σ to fresh constants. The unifier σ' subsumes σ

Given a closed tableau proof $\mathcal{T}$ rooted at Γ, with ground unifier σ, we call abusively the pair $(\mathcal{T}, \sigma)$ a closed tableau, which is ground and without constraint. We refer to tableaux without unifier as *strict/valid tableaux*.

4.1 Origin of the Problem

The naïve translation, that maps inductively each rule of $\mathcal{T}$ to the similar rule of GS3, does not work. Let us translate this way the tableau of Figure 2.

The unifier is $\sigma = \{X := c\}$, and the corresponding GS3 pseudo-proof is the tableau proof simply turned upside down and instantiated, as shown in Figure 4 where bookkeeping contractions have been eluded.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\neg(\exists x(D(x) \Rightarrow \forall y D(y))), \neg(D(c) \Rightarrow \forall y D(y)), D(c), \neg\forall y D(y), \neg D(c) \vdash}\,\text{ax}}{\neg(\exists x(D(x) \Rightarrow \forall y D(y))), \neg(D(c) \Rightarrow \forall y D(y)), D(c), \neg\forall y D(y) \vdash}\,\neg\forall}{\neg(\exists x(D(x) \Rightarrow \forall y D(y))), \neg(D(c) \Rightarrow \forall y D(y)) \vdash}\,\neg\Rightarrow}{\neg(\exists x(D(x) \Rightarrow \forall y D(y))) \vdash}\,\neg\exists$$

Fig. 4: Pseudo sequent derivation for the tableau of Figure 2

The problem in the derivation of Figure 4 is that the $\neg\forall$ rule (the counterpart of the δ rule) requires a *fresh* constant, and it cannot be c, as it was previously introduced by the first $\neg\exists$ rule. In the tableau proof of Figure 2, freshness is innocently masked by the unknown value of X.

The remedy, to show the drinker principle in GS3, is well-known: contract the goal formula, and use once to get a fresh constant c with the $\neg\forall$ rule, and in a *second time* to generate the *same* constant c with the $\neg\exists$ rule.

This is a one-shot particular solution, and we provide below a general jprocedure to treat the problem: given any tableau proof, with a *relaxed* notion of freshness, we force the sequent rules to apply in the *right* order.

4.2 Insight into the translation

Lax freshness is sound for two reasons. First, free variable tableaux are semantically sound. Second, we *syntactically* know it is sound through the unifier σ. The unifiability of the constraints ensure that there is *eventually* no loop. We are in a way guaranteed that there is a right order for the instantiations.

Practice is more subtle. Indeed, any (still naïve) attempt to order all quantifiers of the tableau by a combination of subterm order and precedence in formula[5], topologically sort them to unravel the tableau and get the right order for rules, fails. There is a theoretical argument: free-variable tableaux with on-the-fly Skolemization can be non-elementarily shorter [13, 1, 6] than sequent proofs, namely because of the relaxed notion of freshness, post-checked at unification time. This appears clearly in Figure 4: the two precedence constraints on the $\neg\forall$ and $\neg\exists$ rules are conflicting.

[5] quantifier $\mathcal{Q}_X$ would have priority over $\mathcal{Q}_Y$ if it is higher in the same formula *or* if the instance (by σ) of the metavariable/term introduced by $\mathcal{Q}_Y$ contains the Skolem term introduced by $\mathcal{Q}_X$.

The proofs of the drinker principle gives us a hint: duplication. This removes the above theoretical barrier, as the sequent proof now grows much bigger than the tableau proof. This also means we will make the translated sequent grow *from the root* to its axioms, ensuring at every step soundness (the – open – sequent proof is GS3-valid) and progress (one tableau rule has been considered).

Let us translate the example to have a preview of what we will do. For the sake of readability, and in analogy with the next sections, we let Γ be the root formula $\neg(\exists x(D(x) \Rightarrow \forall y D(y)))$. Translating the first three rules is easy (see Figure 5a). Next, we face the problem discussed above and solve it in four steps:

1. **Save** the current incomplete proof-tree.
2. **Clean** the targeted open leaves: remove all formulas but Γ and the δ formula of interest.
3. **Apply** the now legal δ rule, and clean more (Figure 5b).
4. **Graft** the saved proof-tree 1 to the targeted open leaves (Figure 5c). In fact, make the grafts of step 3 **grow** following the saved proof-tree. **Keep the Skolem formula** as an additional side formula on the relevant branches (in our example: on the single grafted branch).

After those steps, we are able to translate further the tableau, in our case, the sole axiom rule.

Grafting a proof-tree with more than one open leaf multiplies the number of leaves of the tree. Translating a *single* tableau rule into *several* sequent rules is unavoidable, and both height and width grow. So, in general, a single tableau branch (resp. rule) corresponds to several sequent branches (resp. rules). The general mechanism is discussed in the next sections.

4.3 Initial Definitions and Lemmas

We have already mentioned that the GS3 proof is not built by structural induction. We thus need some additional definitions.

Definition 1 (Initial part). *Let T be a closed strict tableau rooted at Γ. An open tableau T_0 is said to be an initial part of T iff it is rooted at Γ and:*

- *either T_0 is a leaf:*
  - *if the root of T is also a leaf (closed by hypothesis), T_0 is a* closed *leaf;*
  - *if the root of T is an internal node, T_0 is an* open *leaf.*
- *or the rule applied at the root of T_0 is exactly the same as the rule applied at the root of T and the sub-tableau(x) of T_0 are initial parts of the corresponding sub-tableau(x) of T.*

We use the same terminology for GS3 *proof-trees.*

Alternatively, if we consider a *sequence of tableaux* used to derive tableau T from its root Γ, then T_0 is an initial part of it if, and only if, there exists at least one such sequence where T_0 appears.

$$\dfrac{\dfrac{\Gamma, \neg(D(c) \Rightarrow \forall y D(y)), D(c), \neg\forall y D(y) \vdash}{\Gamma, \neg(D(c) \Rightarrow \forall y D(y)) \vdash}\,\neg\Rightarrow}{\Gamma \vdash}\,\neg\exists$$

(a) First 3 steps of the translation of Figure 2

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\Gamma, \neg D(c) \vdash}{\Gamma, \neg\forall y D(y), \neg D(c) \vdash}\,\mathrm{w}}{\Gamma, \neg\forall y D(y) \vdash}\,\neg\forall}{\Gamma, \neg(D(c) \Rightarrow \forall y D(y)), D(c), \neg\forall y D(y) \vdash}\,\mathrm{w}}{\Gamma, \neg(D(c) \Rightarrow \forall y D(y)) \vdash}\,\neg\Rightarrow}{\Gamma \vdash}\,\neg\exists$$

(b) Cleaning and applying the δ-rule

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\Gamma, \neg\mathbf{D}(\mathbf{c}), \neg(D(c) \Rightarrow \forall y D(y)), D(c), \neg\forall y D(y) \vdash}{\Gamma, \neg\mathbf{D}(\mathbf{c}), \neg(D(c) \Rightarrow \forall y D(y)) \vdash}\,\neg\exists}{\Gamma, \neg D(c) \vdash}\,\mathrm{w}}{\Gamma, \neg\forall y D(y), D(c) \vdash}\,\neg\forall}{\Gamma, \neg\forall y D(y) \vdash}\,\mathrm{w}}{\Gamma, \neg(D(c) \Rightarrow \forall y D(y)), D(c), \neg\forall y D(y) \vdash}\,\neg\Rightarrow}{\Gamma, \neg(D(c) \Rightarrow \forall y D(y)) \vdash}\,\neg\exists}{\Gamma \vdash}\,\neg\exists$$

(c) Graft and grow

Fig. 5: Solving the drinker problem

An initial part T_0 of T shares the same root, nodes, sequents, branches, constraints, paths and rules as T up to the leaves of T_0. T_0 can also be thought of a labeling of the nodes of T as "seen" and "unseen". For instance, the tableau of Figure 5a is an initial part of the tableau of Figure 2.

The following lemma shows that subsequent definitions are well-formed:

Lemma 1. *Let T_0 be an initial part of a closed strict tableau T, $\mathfrak{b}$ an open leaf of T_0, and $\mathfrak{r}$ the rule applied to the corresponding branch $\mathfrak{b}$ on T. The extension of T_0 by the application of $\mathfrak{r}$ on $\mathfrak{b}$ is also an initial part of T.*

Our goal is to incrementally build a GS3 proof-tree by following the rules of T, given a closed (strict) tableau T with a ground unifier σ. In a sense, we replay the steps that were used to build T, get an initial part T_0, and maintain the invariant that the GS3 proof-tree *maps to* T_0. Note again that a single open-branch of T_0 serves to extend *several* branches of the GS3 proof-tree at the same time. We first define the mapping:

Definition 2 (Partial Link). *Let π_0 be an open* GS3 *proof-tree rooted at Γ and let also $s_1, \cdots, s_n$ be its open leaves, containing respectively the sequents $\Gamma_{s_1} \vdash, \cdots, \Gamma_{s_n} \vdash$.*

Let T_0 be an open strict tableau with open leaves $\mathfrak{b}_1, \cdots, \mathfrak{b}_m$, that respectively containing the set of formulas $\Delta_{\mathfrak{b}_1}, \cdots, \Delta_{\mathfrak{b}_m}$. Let σ be a unifier for T_0.

π_0 is partially linked to (T_0, σ) if, and only if, there exists a partial mapping $\mu : \{s_1, \cdots, s_n\} \mapsto \{\mathfrak{b}_1, \cdots, \mathfrak{b}_m\}$, such that $\sigma\Delta_{\mu(s)} \subseteq \Gamma_s$, when $\mu(s)$ is defined.

We say that the leaf s (of π_0) is linked to the leaf $\mu(s)$ (of T_0), and that the formulas of $\Gamma_s \backslash \sigma\Delta_{\mu(s)}$ are the side formulas *of s.*

This notion is readily extended to describe a partial link to a GS3 proof-tree. In this case, there is no need for an unifier.

Notice that, when $\mu(s_i) = \mu(s_j)$, nothing prevents the side formulas of s_i and s_j to be different. Γ_s is only required to contain the instances by σ of the formulas of $\Delta_{\mu(s)}$.

Notice also that μ is not required to be injective or surjective. Non-injectivity accounts for the fact that a single tableau branch is reflected at more than one place on a GS3 proof-tree. Non-surjectivity of the mapping amounts for the fact that some branches of the original proof may not be reflected in π, in particular when π is bilinked (Definition 4 below). One can check that, in the proof of Theorem 3, the link to θ is not surjective, but the link to π is maintained surjective.

We need the two following refinements over partial links:

Definition 3 (Link). *Let Γ be a set of formula. Let π_0 be a proof-tree linked to a tableau (T_0, σ), and assume that:*

- *π_0 and T_0 are both rooted at Γ,*
- *and the mapping μ is total.*

Then π_0 is said to be linked *to (T_0, σ).*

Definition 4 (Bilink). *We say that π, with open leaves $\{s_1, \cdots, s_n\}$ is bilinked to two GS3 proof-trees θ_0 and θ_1 if, and only if, it is partially linked to θ_0 and to θ_1, and the respective mappings μ_0 and μ_1 verify the disjointness and covering conditions:*

- $\mathsf{Dom}(\mu_0) \cap \mathsf{Dom}(\mu_1) = \emptyset$
- $\mathsf{Dom}(\mu_0) \cup \mathsf{Dom}(\mu_1) = \{s_1, \cdots, s_n\}$

Given a link μ between a GS3 open proof-tree π and an initial part of T, the intention is to apply to all the open leaves $s \in \mu^{-1}(\mathfrak{b}_j)$, the same rule as on $\mathfrak{b}_j$. This is formalized in the next definition:

Definition 5 (Parallel extension). *Let π_0 be a GS3 proof-tree, linked to (T_0, σ) with mapping μ_0, where T_0 is an initial part of a closed strict tableau T with unifier σ. Let T_1 be the extension of T_0 along T on some open leaf $\mathfrak{b}$ with rule r.*

The open proof-tree π_1 of GS3 is called a parallel extension of π_0 along T_1 (by r) if it can be linked to (T_1, σ) such that the mapping μ_1 is equal to μ_0, except on the newly created leaves of π_1, in which case the new leaves are mapped to the corresponding premise leaf(s) of r in T_1.

By abuse of language, this process is called the parallel extension of π along T. The equivalent notion can be defined for two (partially) linked GS3 *proofs-terms and we will use the same terminology.*

In practice, π_1 is built out of π_0 by adding the inference rule r on the suitable leaves. Since this consumes exactly one rule of T, the process of parallel extension eventually stops and generates a GS3 proof-tree. This proof-tree is a sequent proof: all its leaves are closed because they are totally linked to leave themselves closed. The main question is whether this is always possible. The example in Section 4.2 shows that it is not so simple.

4.4 Parallel Extensions

Now we are equipped to describe our algorithm and prove the following theorem:

Theorem 2. *Given any closed tableau T with unifier σ, any initial part T_0, and any* GS3 *proof-tree π_0 linked to (T_0, σ), it is possible to parallely extend π_0 along T.*

Proof. Let $\mathfrak{b}$ be an open leaf of T_0, and r the rule applied to it in T. Let T_1 be the extension of T_0 along T on $\mathfrak{b}$ with rule r. Consider the different cases for r:

- r is an α-rule on a formula A: on each $s_i \in \mu_0^{-1}(\mathfrak{b})$, σA is present on s_i by definition of linkedness, we apply r on it. We link this new proof-tree exactly as the old one, and let μ_1 be defined as:
$$\begin{cases} \mu_1(s_j) &= \mu_9(s_j) \quad \text{for any} s_j \notin \mu_0^{-1}(\mathfrak{b}) \\ \mu_1(s_n.0) &= \mu_0(\mathfrak{b}).0 \quad \text{for any} s_i \in \mu_0^{-1}(\mathfrak{b}) \end{cases}$$
Since both the tableau and the GS3 rules are non-destructive, the invariant $\sigma\Delta_{\mu(s)} \subseteq \Gamma_s$ is maintained.
- r is a γ-rule: we do exactly the same.
- r is a β-rule. We act similarly, except that we have two new open leaves in T_1, $\mathfrak{b}.0$ and $\mathfrak{b}.1$. As well, all the s_i open leaves of π_0 split into $s_i.0$ and $s_i.1$. The new linking function μ_1 is straightforward:
$$\begin{cases} \mu_1(s_j) &= \mu_0(s_j) \quad \text{for any} s_j \notin \mu_0^{-1}(\mathfrak{b}) \\ \mu_1(s_n.0) &= \mu_0(\mathfrak{b}).0 \quad \text{for any} s_i \in \mu_0^{-1}(\mathfrak{b}) \\ \mu_1(s_n.1) &= \mu_0(\mathfrak{b}).1 \quad \text{for any} s_i \in \mu_0^{-1}(\mathfrak{b}) \end{cases}$$
- r is a δ-rule: this is entailed by Theorem 3 below. We postpone this case to the end of Section 4.5.
- r is a closure rule: we apply the axiom rule on each $s_i \in \mu_0^{-1}(\mathfrak{b})$. $\mathfrak{b}$ is now a closed leaf of T_1, and accordingly the s_i are no more open. We thus need restrict the domain of μ_0: $\mu_1 = \mu_{0_{|I}}$, where $I = \{s_j \mid \mu_0(s_j) \neq \mathfrak{b}\}$. □

Notice that the choice of the leaf $\mathfrak{b}$ is not imposed. In order to optimize the translation, it is possible to define some heuristics to choose the branch. As well, for better performances, the heuristics may rearrange, on each path, the order of the rules but the theoretical barrier discussed above will still pop up at some point. This is why we do not insist on optimization here.

4.5 Parallel δ-extensions

The possibility of a δ-extension is made possible by the following theorem:

Theorem 3 (δ-theorem). *Let (T, σ) be a closed tableau. Let Γ its root formulas, $\exists x D(x)$ be a formula of it, on which a δ-rule is applied, generating the Skolem term δ and the formula let $D_\delta = D(\delta)$. We consider the instances by σ of those term and formulas, and call them identically.*

Let θ be an (open) GS3 proof-tree composed only with formulas that appear in (T, σ) (as instances by σ of formula of T), rooted at Γ and such that each leaf contains at least Γ.

Assume that a set of leaves, denoted $\mathcal{B}$, contains $\exists x D(x)$. Let π_0 be an initial part of θ.

Then it is possible to build a proof-tree π_1, rooted at Γ, that is bilinked to π_0 and θ with mappings μ_{π_0} and μ_θ respectively, such that:

- *There is no s_1 such that $\mu_\theta(s_1) \in \mathcal{B}$, i.e. the leaves of θ in $\mathcal{B}$ are "unreachable".*
- *for any leaf s_1, such that $\mu_{\pi_0}(s_1)$ is a prefix of a path $\mathfrak{b} \in \mathcal{B}$ (for short: $\mu_{\pi_0}(s_1)$ is a prefix of $\mathcal{B}$), D_δ appears on this node as a side formula.*
- *All other leaves s_1 of π_1 have the same formulas than $\mu_{\pi_0}(s_1)$, or than $\mu_\theta(s_1)$.*

Proof. We build π_1 by induction on the pair (size of Skolem term δ, size of π_0).

First of all, if π_0 has no rule, there is a tension between the imposed formulas at the root of π_0, Γ, and the leaves of π_1 linked to a prefix of $\mathcal{B}$, that contain (at least) Γ, D_δ. That prevents π_1 to be π_0 itself. Indeed, we start with a manipulated clone of θ and we *graft* Γ, D_δ at the leaves $\mathcal{B}$ of θ, as follows:

- We let π_1 be θ where, to all the leaves $\mathfrak{b} \in \mathcal{B}$ we have weakened to get $\Gamma, \exists x D$, applied the δ-rule to generate D_δ, and weakened once again on $\exists x D$. There is no freshness problem, since Γ does not contain any Skolem term or symbol. π_1 has the same leaves as θ, except for a new set of leaves, which we call $\mathcal{B}^\dagger$. It is composed of the $\mathfrak{b}^\dagger = \mathfrak{b}.0^{k_\mathfrak{b}}$, where $\mathfrak{b} \in \mathcal{B}$ and $k_\mathfrak{b}$ is the necessary number of 0 introduced by the δ-rule and the weakenings. The formulas of the leaves in $\mathcal{B}^\dagger$ are exactly Γ, D_δ.
- We define the bilink in the following way:
  - μ_θ is the partial link from π_1 to θ defined on all the leaves $\mathfrak{b}$ of π_1 that are not member of $\mathcal{B}^\dagger$. It is merely the identity:
  $$\mu_\theta(\mathfrak{b}) = \mathfrak{b} \text{ if } \mathfrak{b} \notin \mathcal{B}^\dagger$$
  - μ_{π_0} is the partial link from π_1 to π_0 defined on $\mathcal{B}^\dagger$. It is the constant 0 function, since π_1 has no rule:
  $$\mu_{\pi_0}(\mathfrak{b}^\dagger) = 0 \text{ if } \mathfrak{b}^\dagger \in \mathcal{B}^\dagger$$

Otherwise, π_1 has at least one rule. Then, we consider any initial part π_0^0 of π_0, that has one rule less and is still an initial part of θ. Let us call π_0^1 the proof-tree produced by the induction hypothesis, with mapping $\mu_{\pi_0}^0$ (resp. μ_θ^0) from π_0^1 to π_0^0 (resp. θ).

To go from π_0^0 to π_0, a rule r is applied on leaf $\mathfrak{b}$. We have the following cases:

– $\mathfrak{b}$ is *not* a prefix of $\mathcal{B}$. we simply copy the rule on each branch s_0^1 of π_0^1 linked to $\mathfrak{b}$, i.e. such that $\mu_{\pi_0}^0(s_0^1) = \mathfrak{b}$. The bilink is formed with an unchanged μ_θ. μ_{π_0} is straightforwardly defined from $\mu_{\pi_0}^0$ as in the proof of Theorem 2.
– $\mathfrak{b}$ is a prefix of $\mathcal{B}$ and r is an α-,β-,γ-rule: we simply copy the rule on each branch s_0^1 of π_0^1 linked to $\mathfrak{b}$, let μ_θ unchanged and let μ_{π_0} be defined from $\mu_{\pi_0}^0$ as in the proof of Theorem 2.
In the case of a branching β-rule, we weaken on D on $s_0^1.0$ (resp. on $s_0^1.1$), if $\mathfrak{b}.0$ (resp. $\mathfrak{b}.1$) is no more a prefix of $\mathcal{B}$. At least one of $\mathfrak{b}.0$ and $\mathfrak{b}.1$ is a prefix of $\mathcal{B}$.
– $\mathfrak{b}$ is a prefix of $\mathcal{B}$, r is a δ-rule and either the Skolem term ε is not comparable to δ for the subterm relation, or it contains δ as a subterm: in this case, we copy the rule as above, since the Skolem term is still fresh.
– $\mathfrak{b}$ is a prefix of $\mathcal{B}$, r is a δ-rule and the Skolem term ε is exactly δ. Since only formulas of T, σ appear, the Skolem formula must be exactly D_δ, otherwise the term would be different. By induction hypothesis on π_0^1 and $\mu_{\pi_0}^0$, $\mathfrak{b}$ already contains D_δ as a side formula. π_0^1 has already the desired form and we let $\pi_1 = \pi_0^1$, $\mu_{\pi_0} = \mu_{\pi_0}^0$ and $\mu_\theta = \mu_\theta^0$.
– $\mathfrak{b}$ is a prefix of $\mathcal{B}$, r is a δ-rule and the Skolem term ε is a strict subterm of δ. Let E_ε be the Skolem formula and $\exists y E$ the quantified formula. We cannot apply the δ-rule on $\exists y E$ because ε is not fresh. As well, we cannot recover freshness by weakening on D_δ, since this loses the invariant.
But, since ε is a strict subterm of δ, we can apply the induction hypothesis on π_0^1, on ε with the formula $\exists y E$, the set of leaves $\mathcal{B}_\mathfrak{b} = {\mu_{\pi_0}^0}^{(-1)}(\mathfrak{b})$ and with π_0^1 as an initial part of itself.
We get a proof-tree, that we call (on purpose) π_1, along with a bilink μ^1, μ^2 to π_0^1 and π_0^1. Let s be a branch of π_1. $\mu^2(s) \notin \mathcal{B}_\mathfrak{b}$, because "no $\mu^2(s)$ can be a prefix of $\mathcal{B}_\mathfrak{b}$", and as we chose π_0^1 as an initial part of itself, being a prefix means being equal. Therefore, if s is linked to a prefix of $\mathcal{B}_\mathfrak{b}$, we must have $\mu^1(s) \in \mathcal{B}_\mathfrak{b}$ and s contains the formulas:
  - E_ε by the very hypothesis of Theorem 3
  - all the formulas of the corresponding branch of $\mathcal{B}_\mathfrak{b}$ by the definition of a partial link, that is to say the formulas of the branch $\mathfrak{b}$, plus the formula D_δ since $\mathfrak{b}$ is a prefix of $\mathcal{B}$.

  Therefore all those branches contain the formulas of the branch $\mathfrak{b}.0$ of π_1, plus the side formula D_δ.
  We now proceed to the definition of the bilink of π_1 with π_0 and θ:
  - $\mu_{\pi_0}(s) = \mathfrak{b}.0$ if $\mu^1(s)$ is defined and belongs to $\mathcal{B}_\mathfrak{b}$, otherwise said if $\mu_{\pi_0}^0(\mu^1(s)) = \mathfrak{b}$.
  - $\mu_{\pi_0}(s) = \mu_{\pi_0}^0([\mu^1 \sqcup \mu^2](s))$ if μ_{π_0} is defined on $[\mu^1 \sqcup \mu^2](s)$ and different of $\mathfrak{b}$. The merge $\sqcup$ is well-defined because of the bilink μ^1, μ^2 is disjoint.
  - $\mu_\theta(s) = \mu_\theta^0([\mu^1 \sqcup \mu^2](s))$ otherwise, which is defined exactly when the two other cases fail.

  We indeed compose the partial link functions, except when it comes to the branch $\mathfrak{b}$. It is easy to see that it is a bilink (Definition 4). Moreover, let us check the conditions of the theorem:

- no leaf s such that $\mu_\theta(s)$ is defined is a prefix of $\mathcal{B}$ because this property holds for μ^0_θ. The leaves s linked to a prefix of $\mathcal{B}$ are either such that $\mu_{\pi_0}(s) = \mathbb{b}.0$ or such that $\mu_{\pi_0}(s) = \mu^0_{\pi_0}([\mu^1 \sqcup \mu^2](s))$.
- the leaves linked to a prefix of $\mathcal{B}$ have D_δ, and only D_δ, as a side formula. In the case $\mu_{\pi_0}(s) = \mu^0_{\pi_0}([\mu^1 \sqcup \mu^2](s))$, this is true by hypothesis on $\mu^0_{\pi_0}$ (it adds exactly D_δ as a side formula) and on μ^1/μ^2, that preserve the formulas, since $\mu^1(s)$ does not belongs to/is not a prefix of (which is the same here) $\mathcal{B}_{\mathbb{b}}$.
  In the case $\mu_{\pi_0}(s) = \mathbb{b}.0$, this property has been checked above.
- all other leaves have the same formulas as the branch they are linked to. This is an inductive property of the partial links $\mu^0_{\pi_0}$, μ^0_θ, μ^1 and μ^2.

As a remark, we can see that, if the partial links μ^1 and $\mu^0_{\pi_0}$ are surjective, then the partial link μ_{π_0} is also surjective. □

We conjecture that we can restrict ourselves, in Theorem 3, to the case of a single rule r that applies on all branches of π_0 that are a prefix of $\mathcal{B}$. In this case, we can apply r on all the leaves that are mapped to a prefix of $\mathcal{B}$ *at once*, that can save us to investigate them one by one.

Notice that considering a *set* of leaves $\mathcal{B}$ is essential to be able to apply induction hypothesis twice. This need comes from the fact that we duplicate parts of the proof, and formulas and rules are duplicated: a single tableau rule can be applied several times, in parallel, in the corresponding sequent proof.

We are now in position to show the remaining case of Theorem 2 dealing with r when it is a δ-rule : let δ be the Skolem term, and D_δ the Skolem formula, after instantiation by σ. We apply Theorem 3 to $\theta = \pi_0$, with $\mathcal{B} = \mu_0^{-1}(\mathbb{b})$, and π_0 as an initial part of θ. Due to the non-destructive nature of GS3, Γ appears on each leaf of θ. We get a proof-tree π_1 bilinked to θ/π_0, that is to say linked to π_0 by $\mu_1 = \mu_\theta \sqcup \mu_0$, where all the branches linked to $\mathcal{B}$ (equivalently such that $\mu_0(\mu_1(s)) = \mathbb{b}$) contain D_δ as a side formula. μ_1 is a link because of the covering condition in Definition 4.

Therefore, we have a link μ from π_1 to (T_1, σ), defined by $\mu(s) = \mu_0(\mu_1(s))$ if $\mu_0(\mu_1(s)) \neq \mathbb{b}$, and $\mu(s) = \mathbb{b}.0$. The parallel δ-extension has succeeded as well. □

Lastly, to show Theorem 1, we need to follow strictly GS3 rules, that is to say replace the Skolem terms by fresh constants on the proof-tree obtained by iterating Theorem 2. Since Skolem term are now *fresh*, this boils down to replacing each Skolem term by a different constant. □

5 Related work and Conclusion

The effect of using optimized versions of Skolemization has been well studied for tableaux methods on classical logic.

The increased efficiency resulting from the use of optimized Skolemization in tableaux methods to handle existential quantifiers has seen a nice body of work. Baaz and Fermüller [1] show how a more efficient $\delta^\star$-rule, which offers

non-elementary speedups in proofs. Even more efficient δ-rules, in terms of potential speedups, are presented by Cantone and Nicolosi Asmundo [6] with the $\delta^{\star^{\star}}$ variant and by Giese and Ahrendt [12] with the Hilbert's symbol based δ^{ε} rule. All these enhanced Skolemization procedures are instances of Cantone's and Nicolosi Asmundo's theoretical framework[7]. These demonstrated speedups can be paralleled to the exponential explosion one might experience when syntactically reconstructing tableaux proofs as ground sequent derivations.

The technique we use in this paper to show a syntactic soundness proof for first-order free variable classical tableaux with Skolemization consists in linking proof-trees to synchronize their simultaneous expansions. We are hopeful this can be extended to handle other δ-variants. The need for grafting various subtrees during the construction of sequent proof, to take into account the relative freshness of the Skolem terms, and the consequent growth in width and breadth confirm that, in presence of free variables and Skolemization, tableaux proofs are necessarily shorter in a non-elementary way [1]. This process can indeed make the size of the sequent proof explodes. Our proof also confirms that semantic arguments are shorter and often clearer, even though syntactic transformations are needed in the context of proof verification.

It has to be noticed that (pre-) outer Skolemization or Skolemization after a prenex normal form transformation would ease *a lot* the soundness proof. Since tableaux do not bear any δ rule, we could translate directly the proof in GS3, and apply Skolem theorem (if $\forall x A(f(x)) \vdash$ then $\forall x \exists y A(y) \vdash$). In particular, the proof-tree does not grow, as there is no speedup in tableaux.

Our result is not specific to sequent calculus, it also readily applies to turn free-variable tableaux with Skolemization into tableaux without free variables, and should generalize gently to other logics. In particular, our next goal is to lift this work to the context of deduction modulo [9] , to de-Skolemize proofs, and obtain proofs checkable by tools such as Coq or Dedukti [4].

The advantage of a syntactic transformation that avoids to appeal to the cut rule, as our, is that it paves the way for a *cut admissibility theorem*. Indeed, from a sequent calculus proof with cuts, we would first get universal validity by (sequent) soundness, then derivability of a tableau proof by completeness, and next, a cut-free sequent-calculus proof by our method. Cut elimination is known since the early days of logic for GS3, this is why switching to other calculi is interesting. In particular, in deduction modulo, cut elimination depends on the chosen rewrite system.

We could also automate the transformation, by writing a program, eventually certifying it in Coq, for instance through a certified programming environment as FoCaLiZe [10].

References

[1] Baaz, M., Fermüller, C.G.: Non-elementary Speedups between Different Versions of Tableaux. In: Baumgartner, P., Hähnle, R., Posegga, J. (eds.) TABLEAUX'95. LNCS (LNAI), vol. 918, pp. 217–230. Springer, St.˜Goar (1995)

[2] Beckert, B., Hähnle, R., Schmitt, P.H.: The Even More Liberalized delta-Rule in Free Variable Semantic Tableaux. In: Proceedings of the Third Kurt Gödel Colloquium on Computational Logic and Proof Theory. pp. 108–119. KGC '93, Springer-Verlag, London, UK (1993)
[3] Beth, E.W.: Semantic entailment and formal derivability. Mededelingen van de Koninklijke Nederlandse Akademie van Wetenschappen, Afdeling Letterkunde 18(13), 309–42 (1955)
[4] Boespflug, M., Carbonneaux, Q., Hermant, O.: The $\lambda\Pi$-calculus modulo as a universal proof language. vol. 878, pp. 28–43. CEUR-WS.org (2012), ceur-ws.org/Vol-878/paper2.pdf
[5] Bonichon, R., Delahaye, D., Doligez, D.: Zenon : An Extensible Automated Theorem Prover Producing Checkable Proofs. In: Dershowitz, N., Voronkov, A. (eds.) LPAR. LNCS, vol. 4790, pp. 151–165. Springer (2007)
[6] Cantone, D., Nicolosi Asmundo, M.: A Further and Effective Liberalization of the delta-Rule in Free Variable Semantic Tableaux. In: Selected Papers from Automated Deduction in Classical and Non-Classical Logics. pp. 109–125. Springer-Verlag, London, UK, UK (2000)
[7] Cantone, D., Nicolosi Asmundo, M.: A Sound Framework for delta-Rule Variants in Free-Variable Semantic Tableaux. J. Autom. Reasoning 38(1-3), 31–56 (2007)
[8] Dowek, G.: Polarized resolution modulo. In: Calude, C.S., Sassone, V. (eds.) IFIP TCS. IFIP, vol. 323, pp. 182–196. Springer (2010)
[9] Dowek, G., Hardin, T., Kirchner, C.: Theorem Proving Modulo. J. Autom. Reasoning 31(1), 33–72 (2003)
[10] Dubois, C., Hardin, T., Donzeau-Gouge, V.: Building certified components within focal. In: Loidl, H.W. (ed.) Trends in Functional Programming. Trends in Functional Programming, vol. 5, pp. 33–48. Intellect (2004)
[11] Fitting, M.: First Order Logic and Automated Theorem Proving. Springer-Verlag, 2nd edn. (1996)
[12] Giese, M., Ahrendt, W.: Hilbert's ε-Terms in Automated Theorem Proving. In: TABLEAUX'99. LNCS, vol. 1617, pp. 171–185. Springer-Verlag, London, UK (1999)
[13] Hähnle, R., Schmitt, P.: The liberalized δ-rule in free variable semantic tableaux. Journal of Automated Reasoning 13(2), 211–221 (1994)
[14] Nonnengart, A., Weidenbach, C.: Computing Small Clause Normal Forms. In: Robinson, A., Voronkov, A. (eds.) Handbook of Automated Reasonning, vol. 1, chap. 6, pp. 336–367. Elsevier Science Publishers B.V. (2001)
[15] Schwichtenberg, H., Troelstra, A.S.: Basic Proof Theory. No. 43 in Cambridge Tracts in Theoretical Computer Science, Cambridge University Press, 2nd edn. (2000)
[16] Smullyan, R.: First-Order Logic. Springer (1968)
[17] Waaler, A., Antonsen, R.: A Free Variable Sequent Calculus with Uniform Variable Splitting. In: Mayer, M.C., Pirri, F. (eds.) TABLEAUX'03. LNCS, vol. 2796, pp. 214–229. Springer (2003)