EPiC Series in Computing

Volume 100, 2024, Pages 246–263

Proceedings of 25th Conference on Logic for Programming, Artificial Intelligence and Reasoning

A Generic Deskolemization Strategy

Johann Rosain[1], Richard Bonichon[2], Julie Cailler[3], and Olivier Hermant[4]

[1] ENS Lyon, Lyon, France
Firstname.Lastname@ens-lyon.fr
[2] O(1) Labs
Firstname.Lastname@o1labs.org
[3] University of Regensburg, Regensburg, Germany
Firstname.Lastname@ur.de
[4] CRI, Mines Paris, PSL University, Paris, France
Firstname.Lastname@minesparis.psl.eu

Abstract

In this paper, we present a general strategy that enables the translation of tableau proofs using different Skolemization rules into machine-checkable proofs. It is part of a framework that enables (i) instantiation of the strategy into algorithms for different sets of tableau rules (e.g., different logics) and (ii) easy soundness proof which relies on the *local extensibility* of user-defined rules. Furthermore, we propose an instantiation of this strategy for first-order tableaux that handles notably *pre-inner* Skolemization rules, which is, as far as the authors know, the first one in the literature. This deskolemization strategy has been implemented in the Goéland [17] automated theorem prover, enabling an export of its proofs to Coq [8] and Lambdapi [2]. Finally, we have evaluated the algorithm performances for *inner* and *pre-inner* Skolemization rules through the certification of proofs from some categories of the TPTP [39] library.

1 Introduction

In a way, automated theorem provers (ATPs) can be seen as oracles that generate a yes/no answer for a given formula. Although some provers attempt to provide a trace, the trustworthiness of their answer often depends solely on the confidence level we have in the respective ATP. Nevertheless, these tools are typically complex, extensive software in constant evolution, comprising thousands of lines of code and employing sophisticated heuristics. Moreover, since they are developed by humans, they are inherently error-prone. In such tools, bugs can be disastrous, causing them to prove non-theorems, consequently compromising the reliability of their answers. There are two ways to fight inconsistencies in ATPs: by fully certifying the kernel of the prover, which is a time-consuming, arduous, and long-term work [37], or by producing machine-checkable proofs, which is, as we show in this paper, easily accessible.

The latter relies on the notion of proof certificates. These are proofs generated by an automated theorem prover that an external proof checker can verify. Indeed, in contrast with ATP, proof assistants rely on a small trusted kernel, that boosts confidence in its correctness.

Therefore, it is natural to seek a way to combine the strengths of both worlds by producing checkable proofs, thereby instilling full confidence in the results of the ATP.

A significant strength of tableau-based tools is their ability to produce a proof from the pristine input formula. In particular, non-clausal ground first-order tableaux implement rules that mirror the ones of the sequent system GS3 [40], which can be easily implemented in proof assistants. Thus, by translating this kind of tableau proofs into GS3 sequent proof trees, we can reasonably expect them to produce machine-checkable certificates.

However, due to the use of free variables and the related Skolemization optimizations, a free-variable tableau proof-search procedure can produce a proof slightly different from the one obtained by the usual (ground) tableau calculus, making it not trivially translatable into a sequent system. Indeed, most proof assistants do not accept advanced Skolemization strategies such as those implemented in ATP, that keep correctness but greatly shorten the length of a proof, thus improving its performance. It results in a need for translating free-variable tableau proofs into sequent proofs.

A deskolemization strategy for tableaux has previously been proposed [14] but it is impractical as it always leads to an exponential explosion of the number of branches, and handles Skolemization strategies only up to the quite weak inner Skolemization.

This paper aims to provide a strategy for translating tableau proofs, that use optimized Skolemization techniques, into GS3 sequent proof trees to allow machine certification. Its contribution is many-fold:

- a general strategy able to deskolemize proofs that use different Skolemization rules,
- a soundness proof for this strategy, stating that any algorithm instantiated from this strategy using user-defined *strategy rules* merely needs those to satisfy *local extensibility*,
- an instantiation of this strategy into an algorithm for non-clausal first-order tableaux,
- an implementation and an assessment of this algorithm in the Goéland [17] theorem prover.

Related Work. Certification of ATP proofs has been explored for various techniques, such as deskolemizing the resolution method [27]. For sequent-based systems, such frameworks have been proposed [21, 35], but only a few focus on the certification of tableaux proofs through a translation to GS3 [12, 14]. The former require side information and is based on connection tableaux proofs which alter the input formula, while the latter proposes an algorithm to translate on the fly first-order tableaux proofs that use inner Skolemization rules. Our approach provides an algorithm that takes the best of these last two work: an on-the-fly translation of tableaux proofs that deskolemizes as parsimoniously as possible. In a complementary way, [19] has proposed a framework to verify the soundness of Skolemization rules.

2 Context and Preliminary Definitions

We work in the setting of first-order logic [41]. A signature is a tuple $\Sigma = (\Sigma^F, \Sigma^P)$ consisting of a set Σ^F of function symbols and Σ^P of predicate symbols, each symbol having a fixed arity. For the purposes of Skolemization, we assume Σ^F to be countable for any arity. Given a signature Σ, we define the usual notions of Σ-terms, Σ-atoms, Σ-literals and Σ-formulas, as well as the application of a substitution σ over a formula F or a term t, that we denote $\sigma(F)$ or $\sigma(t)$. The set of *free variables* (i.e., variables that are not under the scope of a quantifier) for a formula F is denoted $\mathrm{FV}(F)$. Moreover, we say that $t \in F$ or $t \in t'$ if there exists a position ω such that $F_{|\omega} = t$ or $t'_{|\omega} = t$.

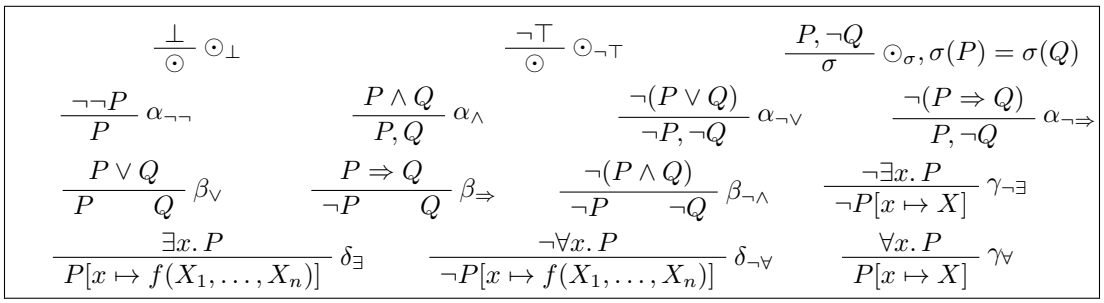

Figure 1: Rules of the free-variable tableaux calculus.

Free-Variable Tableaux. The free-variable tableau [28] method is a variant of the original ground calculus [11,33] that allows the use of free variables in place of ground terms, in order to delay instantiation and improve the performances of ATP. Throughout the paper, we assume a standard (refutationally complete) set of first-order inference rules categorized in α rules (unary inferences), β rules (binary inferences), γ rules (unary rules that introduce free variables) and δ rules (unary rules introducing Skolem symbols). We say that F is a *δ-formula* (respectively *γ-formula*) if a δ rule (respectively γ rule) can be applied to it. If D is a δ-formula, then δ_D is the term (called δ-term) generated by applying the δ rule on D. The free-variable tableau rules are presented in Fig. 1, and illustrated by a proof in Fig. 3a.

A *tableau* (T, σ) for a formula F is a pair of a tree T whose nodes are decorated with sets of formulas, and a substitution σ over the free variables of these formulas, such that either (i) T is a single-node tree, and σ is the empty substitution, or (ii) T is obtained by application of an inference rule on a tableau (T', σ'). A branch in a tableau T is said to be *closed* if it contains two literals A and B such that $\sigma(A) = \sigma(\neg B)$. A tableau is closed if all its branches are closed and thus represents a *proof* for F, that has to be read from top to bottom. In this paper, we mostly consider *closed tableaux*, and thus we often say *tableau* instead of *closed tableau*. We denote $F \rhd F'$ if F' is derived from F using a tableau rule, $\rhd^+$ the transitive closure of this relation, and $\rhd^*$ the reflexive closure of $\rhd^+$. Furthermore, given N a node of a tableau, the set of formulas labeling N is denoted $\mathcal{L}(N)$. We may (purposefully) make the confusion between "node" and "branch" as these notions are isomorphic in nondestructive tableaux. As such, for a branch B, we write $\mathcal{L}(B)$ for the label of the last node of B.

Skolemization. Skolemization consists in eliminating existential quantifiers and replacing the quantified variable with a function symbol with parameters, under specific constraints. In first-order tableaux, Skolemization is defined by the two δ rules of Fig. 1. Among the various Skolemization strategies presented in the literature [4, 9, 18, 28, 29, 31], we consider only *outer* (δ) [28], *inner* (δ^+) [1, 31] and *pre-inner* (δ^{+^+}) [9] Skolemization in this paper. All these rules impose the function symbol to be fresh. The outer Skolemization technique sets $X_1, \ldots, X_n$ to be all the free variable of the branch of F. In inner Skolemization, $\{X_1, \ldots, X_n\} = \mathrm{FV}(F)$. Finally, pre-inner Skolemization relaxes the inner Skolemization freshness condition for f, by sharing the same unique symbol $f_{[F]}$ between all δ-formulas α-equivalent to F.

GS3 Calculus. The Gentzen-Schütte calculus (GS3) [40] is the sequent system that has inspired tableaux. Its rules mirror those of the (ground) tableau calculus, which does not make use of free variables. Fig. 2 presents the GS3 version of the rules that differ from those introduced

$$\frac{\Delta, \exists x.\, P, P[x \mapsto c] \vdash}{\Delta, \exists x.\, P \vdash}\ \exists \qquad \frac{\Delta, \neg\forall x.\, P, \neg P[x \mapsto c] \vdash}{\Delta, \neg\forall x.\, P \vdash}\ \neg\forall$$

$$\frac{\Delta, \forall x.\, P, P[x \mapsto t] \vdash}{\Delta, \forall x.\, P \vdash}\ \forall \qquad \frac{\Delta, \neg\exists x.\, P, \neg P[x \mapsto t] \vdash}{\Delta, \neg\exists x.\, P \vdash}\ \neg\exists$$

Figure 2: Rules of the GS3 calculus that differ from the tableau calculus.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\neg(\exists x.D(x) \Rightarrow \forall y D(y))}{\neg(D(X) \Rightarrow \forall y D(y))}\,\gamma_{\neg\exists}}{D(X), \neg(\forall y D(y))}\,\alpha_{\neg\Rightarrow}}{\neg D(f(X))}\,\delta_{\neg\forall}}{\neg(D(f(X)) \Rightarrow \forall y D(y))}\,\gamma_{\neg\exists}}{D(f(X)), \neg\forall y D(y)}\,\alpha_{\neg\Rightarrow}}{\odot}\,\odot$$

(a) Outer Skolemization tableau.

$$\dfrac{\dfrac{\dfrac{\dfrac{\neg(\exists x.D(x) \Rightarrow \forall y D(y))}{\neg(D(X) \Rightarrow \forall y D(y))}\,\gamma_{\neg\exists}}{D(X), \neg(\forall y D(y))}\,\alpha_{\neg\Rightarrow}}{\neg D(c)}\,\delta^{+}_{\neg\forall}}{\{X \mapsto c\}}\,\odot_{\sigma}$$

(b) Inner Skolemization tableau.

Figure 3: Proof of the drinker paradox in outer and inner Skolemization.

in Fig. 1. Contrarily to tableaux rules, these read from bottom to top. The notion of tableaux node and branch extends readily to GS3. Moreover, the derivation relation for tableaux and for GS3 proofs is similarly denoted by $\triangleright$.

Proof-Tree Manipulations. Let us define the different notions to characterize proof trees. First, we call *illegal δ-term* a term that does not respect the freshness condition of the GS3 rules on existential formulas and impacts the *well-formedness* of the proof tree.

Definition 1 (Illegal δ-Term)**.** *Given a branch B of a GS3 proof tree, a δ-term δ_D is said to be illegal iff the node where it is created already contains δ_D.*

Definition 2 (Well-formed Proof Trees)**.** *A GS3 proof tree π is well-formed when every node of π is an instance of the GS3 calculus and no illegal δ-term appears. A GS3 proof π is* well-formed *if it is a well-formed proof tree and all leaves are axiom rules.*

Definition 3 (Initial Part)**.** *Let (T, σ) be a tableau proof. (T_0, σ_0) is an initial part of (T, σ) if, and only if, T_0 and T share the same root and either the root is a leaf of T_0, or the same rule is applied to the common root of T_0 and T, and the children of (T_0, σ_0) are initial parts of the corresponding children in (T, σ).*

We also define *leaves* and *initial segments*, a *leaf* being the last node of a branch, with its corresponding set of formulas, and an *initial segment* is the analog of an initial part but for the branches, denoted $B \sqsubseteq B'$ for "B is an initial segment of B'". Those three notions seamlessly extend to GS3 proof trees. Now, we formally define the notion of *mapping*, which represents a correspondence function between a tableau and a sequent proof tree, as well as an order relation over these mappings. The idea is, starting from a branch in a GS3 proof tree, to find the "equivalent branch" in the tableau and track its evolution w.r.t. this branch.

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

Figure 4: Incorrect proof yielded by a naive translation in GS3 of the tableau proof of the drinker paradox in inner skolemization.

Definition 4 (Mapping)**.** *Let (T, σ) be a tableau proof tree and π a GS3 proof tree. A mapping $\mu : \pi \to \sigma(T)$ is a function that associates a branch $B \in \pi$ to a branch in $\sigma(T)$ such that $\mathcal{L}(B) \supseteq \mathcal{L}(\mu(B))$.*

The notion of mapping between a GS3 proof tree and a tableau can also be extended to be between two GS3 proof trees (without the need for the substitution). Sequent–sequent mappings will be denoted by the symbol λ, while sequent–tableau mappings will be denoted by μ. We now specify an order over mappings, which will be used to prove that (i) the sequent proof tree built by our strategy eventually reaches a more advanced state (ii) there is no regression while applying it, and thus that (iii) our strategy terminates. To achieve this, we consider a multiset ordering [23] over the mappings image without multiplicity, so that (ii) holds even in the steps of our process that involves branch duplication.

Definition 5 (Mapping Ordering)**.** *Let μ and μ' two mappings from sequents π and π' to a tableau T. We say that $\mu' \leq \mu$ iff forall $B' \in \pi'$, there exists $B \in \pi$ such that $\mu(B) \sqsubseteq \mu'(B')$.*

Example 1. *Let B, C be two incomparable branches (w.r.t. the segment extension relation $\sqsubseteq$) of a tableau T, and B_1 and B_2 branches of T such that B is a strict initial segment of B_1 and B_2 : $B \sqsubset B_i$. Let us consider three mappings μ_1, μ_2, and μ_3 with respective images $\{C, C, B, B, B_1, B_2\}$, $\{C, C, B, B, B, B_1\}$, and $\{C, B, B\}$. We have $\mu_1 \leq \mu_2 \leq \mu_3 \leq \mu_1$. Note that $\{B\}$ is incomparable with $\{C, B_1, B_2\}$.*

Need for a Translation. A proof in standard tableaux such as Fig. 3a is equivalent to a ground tableau [28] by a mere replacement of δ-terms with fresh constants throughout the proof tree. Therefore it can also be turned into a GS3 proof, making the translation from tableaux to GS3 a simple one-to-one mapping. Proofs using optimized Skolemization strategies are shorter and thus lose this trivial mapping, the crux of the problem being the translation of δ rules. For instance, a proof using δ^+ rules is developed in Fig. 3b. It is shorter than the proof of Fig. 3a, and the naive translation attempt of Fig. 4 fails when the rule denoted ($\star$) is applied: an *illegal δ-term* appears. Changing the name of the δ-term to c' do not solve the issue, since this breaks the axiom rule.

3 The Deskolemization Strategy

In this section, we present a strategy that overcomes the limitations raised in the previous section by offering an on-the-fly translation that relies on the principle of *deskolemization*. It is a refinement of both [12, 14], but keeps a high theoretical complexity as deskolemizing proofs is

non-trivial [3,5]. However, this improved version (i) generalizes well to Skolemization strategies that fulfill weak conditions (c.f. Sec. 5) and (ii) behaves well in practice (c.f. Sec. 6).

Our idea is, given a tableau proof (T, σ), to build a GS3 proof by following the rules executed from the root of T until reaching a δ rule that would introduce an illegal δ-term. Then, we remove the formulas in which this term appears (by applying *weakening* steps), apply the now-legal δ rule and grow back the proof tree to get the pre-removal sequent leaves back with, in addition, the skolemized formula. To efficiently select which formulas to keep or remove, we introduce the notions of *dependent* formulas (adapted from [12]) and *descendant* formulas.

Definition 6 (Dependency)**.** *Let N be a node of a GS3 proof tree. Let, in $\mathcal{L}(N)$, D be a δ-formula and F be a γ-formula. F depends on D if and only if δ_D does not appear in F, $F \rhd F'$, and $\delta_D \in F'$. The set of formulas which depend on D is denoted $\Delta(D)$:*

$$\Delta(D) = \{F \in \mathcal{L}(N)_\gamma \mid \delta_D \notin F \wedge F \rhd F' \wedge \delta_D \in F'\}$$

This set captures all the γ-formulas that instantiate a quantified variable with δ_D as a subterm. In addition, as δ_D does not appear in any dependent formula, $\Delta(D)$ gives a starting point to the algorithm to subsequently select the formulas that need to be weakened once an illegal δ-term is identified — the descendants of the dependent formulas. We call G a *descendant* of a formula F if $F \rhd^* G$. Moreover, if $F \rhd G$ in one step, G is the *direct descendant* of F. Note that there may be multiple direct descendants of a single formula.

Definition 7 (Dependent Descendance)**.** *Let F and D such that $F \in \Delta(D)$. The set of formulas descending from F which are also dependent on D is denoted $\Lambda(F, D)$ and defined as:*

$$\Lambda(F, D) = \left\{G \in \mathcal{L}(L) \mid F \rhd^+ G \wedge \forall F_i.(F \rhd^+ F_i \rhd^* G) \Rightarrow \delta_D \in F_i\right\}$$

In order to unequivocally identify the γ-formula that is the source of the dependency, Def. 7 imposes the δ-term to continuously appear in the descendance chain. In practice, this condition is not mandatory.

We can now define our deskolemization strategy.

1. If the rule applied is not a δ rule, apply the corresponding GS3 rule (Fig. 5a).
2. Otherwise, let D be the active δ-formula. If the δ rule has already been applied on D and its direct descendant is still on the node, then go back to Step 1. Else, for every formula F in $\Delta(D)$, weaken the sequent to remove all formulas of $\Lambda(F, D)$ and record the rules used to derive these formulas in their application order in a list called $\mathcal{R}$ (Fig. 5b).
3. Apply the δ rule on D (first step of Fig. 5c).
4. Apply back the rules recorded in $\mathcal{R}$, using the corresponding *strategy rules* (last two steps of Fig. 5c).
5. If a leaf has not been reached, go back to Step 1.

Deskolemization algorithms are then *instances* of this strategy that provide a *strategy rule* to specify the behavior that should be adopted during the reapplication phase of Step 4.

Definition 8 (Strategy Rule)**.** *A* strategy rule *is an algorithm $\mathcal{SR}(\pi, B, \pi_0, \lambda, F, \partial)$ where:*

- *π is a GS3 proof tree during the reapplication phase (i.e., Step 4, after the weakening and the application of a δ-rule),*

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

(a) First steps of the proof.

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

(b) Cleaning the relevant formulas descending from $\Delta(D)$.

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

(c) Skolemization, applying back $\mathcal{R}$'s rules and finalisation.

Figure 5: Sound translation into GS3 of the drinker paradox in inner Skolemization.

- *B is the branch of π where the reapplication is performed,*
- *π_0 is the initial part of π where the initial segment of B is B before Step 2 and all the others branches of π_0 are in π,*
- *λ is a mapping from π to π_0 that is minimal w.r.t. $\leq$,*
- *F is the formula in B on which the rule ∂ should be applied,*

and returns a triple (π', B', λ') where π' is an extension of π where ∂ has been applied on F in B, yielding the branch B' and λ' is a mapping that extends λ.

We now focus on defining a framework and, in particular, the notion of *local extensibility* that will specify the necessary conditions that a strategy rule needs to fulfill in order to yield a sound and terminating procedure.

4 A Soundness Framework for the Strategy

The idea of the soundness proof is to build and maintain a *mapping* between an under-construction GS3 proof tree and a given tableau proof. We prove that the mapping grows throughout the application of the deskolemization strategy and terminates, leading to a well-formed GS3 proof tree. The goal of this framework is to provide a general-purpose proof of (i) soundness for any procedure implementing the strategy and (ii) termination, provided *local extensibility* of the strategy rules introduced.

Let us start by introducing the key notion of *relevant formula*. These formulas are the ones that (legally) introduce δ-terms required to close a branch — either by being directly involved in the closure, either by being needed by a formula involved in the closure. For instance, when $\delta_\exists(d)$ is applied in π_2 of Fig. 6, both $P(c)$ and $Q(d)$ are *relevant formulas*, as weakening one would lead to replay the associated δ rule and looping infinitely over the same tree.

Definition 9 (Relevant Formula)**.** *Let F, F' be formulas of an under-construction GS3 proof tree π such that F is δ-formula and $F \rhd F'$ in a prior proof step S.*

We say that F' is relevant *in a node N if $F' \in N$ and, by applying the deskolemization strategy, S is supposed to occur at least once in the subtree rooted at N.*

This definition takes place in the context of the reapplication process. We want to characterize the formulas resulting from the application of a δ rule that have to be kept in the branch and not weakened anymore. However, identifying such formulas is not trivial: for instance, in Fig. 6, if $Q(x)$ is replaced by $Q(y, x)$, then the generated formula $Q(c, f(c))$ becomes relevant in π_1 *before* introducing $P(c)$. Introducing the latter, while at the same time requiring to keep $Q(c, f(c))$, results in an unplanned weakening, thus leading the strategy to fail as it will not be able to find the formula that has introduced $Q(c, f(c))$. To take this into account, we show that keeping only the *exclusive* formulas is sufficient to generate all the relevant formulas of a branch. In the following definitions, we focus on formulas that have introduced a δ-term, i.e., on direct descendants of δ-formulas.

Definition 10 (Exclusive δ-Terms)**.** *Let δ_0, δ_1 be two δ-terms. δ_0 excludes δ_1 if $\delta_1 \notin \delta_0$.*

Example 2. *c and d are mutually exclusive. Moreover, c excludes c and $f(c)$. But $f(c)$ does not exclude c.*

Definition 11 (Exclusive Formula)**.** *Let F_δ and G_δ be δ-formulas, with $F_\delta \rhd F$ and $G_\delta \rhd G$. F excludes G if δ_F excludes δ_G, i.e., if $\delta_G \notin \delta_F$.*

Example 3. *$P(c)$ and $Q(d)$ are (mutually) exclusive formulas. Furthermore, $P(c)$ excludes $Q(c, f(c))$. But $Q(c, f(c))$ does not exclude $P(c)$.*

Remark 12. *The notions of* non-exclusion *is irrelevant between formulas coming from the same branch in the original tableau proof. Indeed, the notion of* dependency *already manages same-branch non-exclusive formulas. Thus, we will say that F does not exclude G only if F and G do not originate from the same tableau branch.*

We say that a formula F is exclusive *to a node N* if for all δ-formula G of N such that $G \rhd G'$ and $G' \notin N$, F excludes G'. Moreover, we say that F is exclusive *to a branch rooted in N* if it is exclusive to N and all its children. This allows us to characterize formulas that are exclusive to the remaining unprocessed δ-formulas. This definition captures the idea of F not being dependent of any δ-term that has to be generated in a subsequent step by a formula currently in the node (resp. branch).

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\forall x, y.(\neg P(y) \wedge \exists z.P(z)) \vee (\neg Q(x) \wedge \exists z.Q(z))}{(\neg P(Y) \wedge \exists z.P(z)) \vee (\neg Q(X) \wedge \exists z.Q(z))}\,\gamma_\forall(X), \gamma_\forall(Y)}{\neg P(Y) \wedge \exists z.P(z)}\,\beta_\vee}{\neg P(Y), \exists z.P(z)}\,\alpha_\wedge}{P(c)}\,\delta_\exists(c)}{\{Y \mapsto c\}}\,\odot \qquad \dfrac{\dfrac{\dfrac{\neg Q(X) \wedge \exists z.Q(z)}{\neg Q(X), \exists z.Q(z)}\,\alpha_\wedge}{Q(d)}\,\delta_\exists(d)}{\{X \mapsto d\}}\,\odot$$

(a) Tableau proof

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\odot\,\dfrac{}{\cdots, P(c), \cdots, \neg P(c), \exists z.P(z) \vdash}}{\cdots, P(c), \cdots, \neg P(c) \wedge \exists z.P(z) \vdash}\,\alpha_\wedge^{3,4} \qquad \pi_2}{\cdots, P(c), \cdots, (\neg P(c) \wedge \exists z.P(z)) \vee (\neg Q(d) \wedge \exists z.Q(z)) \vdash}\,\beta_\vee^{2,3}}{\cdots, P(c) \vdash}\,\gamma_\forall^2(c)}{\cdots, \forall y.(\neg P(y) \wedge \exists z.P(z)) \vee (\neg Q(d) \wedge \exists z.Q(z)), \exists z.P(z) \vdash}\,\delta_\exists(c)}{}\,\mathrm{w}[\gamma_\forall^1(c), \beta_\vee^1, \alpha_\wedge^1] \qquad \pi_1}{\cdots, \neg P(c), \exists z.P(z) \vdash}\,\alpha_\wedge^{1,2}}{\cdots, \neg P(c) \wedge \exists z.P(z) \vdash}\,\beta_\vee^1}{\cdots, (\neg P(c) \wedge \exists z.P(z)) \vee (\neg Q(d) \wedge \exists z.Q(z)) \vdash}\,\gamma_\forall^1(d), \gamma_\forall^1(c)}{\forall x, y.(\neg P(y) \wedge \exists z.P(z)) \vee (\neg Q(x) \wedge \exists z.Q(z)) \vdash}$$

(b) The GS3 proof-tree after processing $\delta_\exists(c)$ (and the associated closing rule)

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\odot\,\dfrac{}{\cdots, P(c), \cdots, \neg P(c), \exists z.P(z) \vdash}}{\cdots, P(c), \cdots, \neg P(c) \wedge \exists z.P(z) \vdash}\,\alpha_\wedge^{7,8}}{\cdots, P(c), \cdots, Q(d), \cdots, \neg P(c) \wedge \exists z.P(z) \vdash}\,\mathrm{w}[\alpha_\wedge^6] \qquad \alpha_\wedge^{9,10}\,\dfrac{\odot\,\dfrac{}{\cdots, Q(d), \cdots, \neg Q(d), \exists z.Q(z) \vdash}}{\cdots, P(c), \cdots, Q(d), \cdots, \neg Q(d) \wedge \exists z.Q(z) \vdash}}{\cdots, P(c), \cdots, (\neg P(c) \wedge \exists z.P(z)) \vee (\neg Q(d) \wedge \exists z.Q(z)) \vdash}\,\beta_\vee^{4,5}}{\cdots, P(c), \cdots, Q(d) \vdash}\,\gamma_\forall^2(d), \gamma_\forall^3(c)}{\cdots, P(c), \cdots, \exists z.Q(z) \vdash}\,\delta_\exists(d)}{}\,\mathrm{w}[\gamma_\forall^1(d), \gamma_\forall^2(c), \beta_\vee^3, \alpha_\wedge^5]}{\cdots, P(c), \cdots, \neg Q(d) \wedge \exists z.Q(z) \vdash}\,\alpha_\wedge^{5,6}}{\cdots, \exists z.P(z), P(c), \neg Q(d) \wedge \exists z.Q(z) \vdash}\,\mathrm{w}[\alpha_\wedge^2]$$

(c) The GS3 proof-tree of π_2 after processing of $\delta_\exists(d)$ (and the associated closing rule)

Figure 6: An example with the relevant formula $P(c)$ highlighted throughout its life cycle in the proof tree. $Q(d)$ is also a relevant formula in π_2 (and therefore in π_1).

Lemma 13. *Exclusive formulas will never be weakened by the strategy.*

Proof. Suppose that an exclusive formula F is weakened by the strategy. Then it means that there exists a δ-term δ_0 that triggers the reapplication mechanism. As F is weakened, it means that $\delta_0 \in F$. Furthermore, $\delta_0 \in \delta_F$ as it was (necessarily) introduced by a γ rule. Thus, F is not exclusive to a formula in the branch which is a contradiction of the assumption. □

Exclusivity is a *total* relation: either F excludes G, or G excludes F. Indeed, if F and G

are mutually non-exclusive, by definition $\delta_F \in \delta_G$ and $\delta_G \in \delta_F$, which is impossible.

Definition 14 (Critical Node)**.** *Let N be a node of a GS3 proof tree such that a rule that introduces an illegal δ-term δ_D should be applied (before Step 2). Let N' be the descendant of N such that all formulas that depend on D have been weakened and where the δ rule is applied (after Step 3). N' is a* critical node *for D.*

Lemma 15 (Relevant Formulas are Exclusive)**.** *Let N be a node of a GS3 proof tree. If N is a critical node for a formula D, then all the relevant formulas of N exclude D.*

Proof. By contradiction. Assume that F is a relevant formula for N that does not exclude D. By definition, $\delta_D \in \delta_F$ and $F \in \Delta(D)$. But, as N is a critical node, all the formulas that depend on δ_D have been weakened. □

We can now refine the notion of *relevant formula* applied to a branch: a formula F is said to be relevant for a branch B if it is relevant to at least one critical node of B.

Corollary 16. *Given a branch B rooted in a node N of a GS3 proof tree, not weakening the exclusive formulas of B is sufficient to generate all the the relevant formulas of B.*

Proof. We prove that the set of relevant formulas of a branch B is a subset of the exclusive formulas of B. There are two cases.

- Either B has no critical node, i.e., every δ term generated has an empty dependency set. Then, all relevant formulas are exclusive, otherwise F is non-exclusive to some D that introduces δ_D and, by definition, $\delta_D \in \delta_F$ and at least one node becomes critical.
- or B has critical nodes, and by Lem. 15, all relevant formulas of such nodes are exclusive.

□

We have shown that generating and keeping the exclusive formulas of a branch eventually generates all the relevant formulas of this branch. This is a key property of the method, as if all the relevant formulas are present in a branch, all their subbranches will close trivially: if the reapplication routine is subsequently called, no other branch can lead back to the current branch as no δ formula that triggered this behavior will be reapplied.

We can now define the *local extensibility* conditions that the strategy rules need to satisfy in order to have a sound and terminating strategy. It introduces five requirements: one about the minimality of the mapping between the sequent proof trees yielded, one that ensures the non-regression of this mapping, one that enforces the progression of the "focused" branch and two that are necessary to produce a well-formed finite proof tree. The first allows us to recover the mapping to the tableau when the routine halts. The second certifies that only B and its extensions are actually updated in the mapping. The third prevents from stagnation by forcing the application of a rule. The penultimate makes sure that no illegal δ-terms are introduced. The last one ensures the overall termination of the strategy.

Definition 17 (Local Extensibility (Fig. 7))**.** *Let $\mathcal{SR}$ be a* strategy rule*. $\mathcal{SR}$ is* locally extensible *if $\mathcal{SR}(\pi, B, \pi_0, \lambda, F, \partial)$ returns (π', B', λ') such that:*

- *λ' is minimal w.r.t. $\leq$,*
- *$\lambda' \leq \lambda$,*
- *$\lambda(B) \sqsubset \lambda'(B')$,*
- *π' is well-formed whenever π is well-formed,*

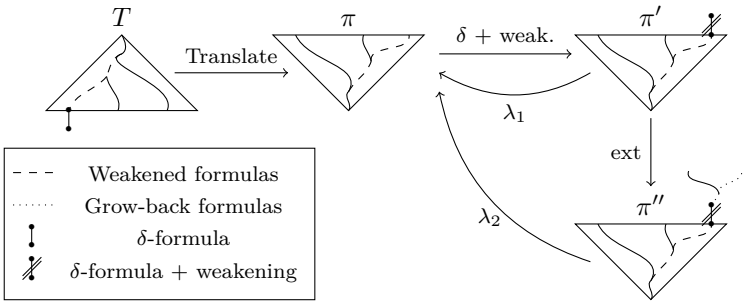

Figure 7: Local extensibility of a strategy rule applied on a GS3 proof tree π mapped to a tableau T. The growing-back phase starts by weakening and applying the δ rule on π. Then, the weakened formulas are regenerated by reapplying missing rules, resulting in the sequent proof tree π'', which is mapped to T after the application of the δ rule through the intermediate λ_i, mappings to π that grow with the replayed rules.

- *no exclusive formula is weakened in π'.*

To give an insight, the algorithm maintains a mapping between an initial part of the tableau proof and a GS3 proof tree under construction. This tableau mapping can be locally lost during the reapplication phase, but we need to provide constant progress when applying back the weakened rules. We thus consider an intermediate mapping during the grow-back phase, in which the sequent proof tree is mapped to an initial part of itself before the weakening phase. We show that the application of the strategy rule makes the mapping decrease, i.e., makes the current sequent proof tree being mapped to a (strictly) more advanced initial GS3 proof tree. Hence, we show that the reapplication of the rules preserves the mapping and can only *extend* it by adding new elements without removing anything or altering other branches.

A δ-term δ_D is said *frozen* [25] (a more generic definition can be found in [35, Def. 6.1]) if all the variables of δ_D are free variables of the propositions where δ_D appears. In tableaux, all δ-terms are obviously frozen, and it has been shown [25] that replacing frozen terms by fresh constants keeps proofs well-formed. Hence, we consider a δ-term $f_D(t_1, \ldots, t_n)$ to be *unique*. It thus allows us to seamlessly consider the δ-term δ_D as either a tableau term or a GS3 constant.

With these definitions in place, we now prove that our strategy terminates and produces a *sound* translation. We start by avoiding trivial non-termination by requiring a (weak) hypothesis on the δ rules, that entails that a δ rule cannot depend on itself.

Lemma 18 (No Self Dependency)**.** *Let D be a δ-formula. If the term δ_D generated by skolemizing D contains all the free variables of D, then for all $F \in \Delta(D)$, $D \notin \Lambda(F, D)$.*

Proof. Assume that there exists $F \in \Delta(D)$ such that $D \in \Lambda(F, D)$. It means that $\delta_D = f_D(t_1, \ldots, t_n) \in D$. This term contains the Skolem function symbol f_D, that is absent from the root of the tableau, so it has been introduced by an ancestor of D :

- by a γ rule. By definition of the δ rule, $\delta_D \in \delta_D$. It creates an infinite term, which cannot happen in first-order logic.
- by a δ rule. By definition, f_D needs to be at least *specific* to both this ancestor and D. But they cannot be α-equivalent, so it cannot happen by assumption: the reference tableau proof is sound.

□

We will now focus on the notion of mapping by showing that (i) there exists a mapping (ii) that does not regress (iii) and that strictly extends a branch by integrating a tableau rule in the GS3 proof tree.

Lemma 19 (Mapping Progression)**.** *Let $\mu : \pi \to \sigma(T)$ a mapping. If a rule ∂ of T has to be integrated in π on the branch B, then there exists π', $B' \in \pi'$ such that $B \sqsubset B'$, and a mapping $\mu' : \pi' \to T$ such that $\mu' \leq \mu$ and $\mu(B) \sqsubset \mu'(B')$.*

Proof. By case analysis. Let L be the leaf of B in π and F the formula on which ∂ should be applied. We will identify the nodes of proof trees using their labels.

- ∂ is an α or a γ rule and let F' be such that $F \rhd F'$. Let $L' = L \cup \{F'\}$, which is the leaf of B' ($\sqsupset B$) in π'. As F' is the only addition to $\mu(L)$ in T, the mapping $\mu' : \pi' \to \sigma(T)$ is defined as follows for every branch b of π':

$$\mu'(b) = \begin{cases} \mu(B) \cup \{F'\} & \text{if } b \text{ is } B' \\ \mu(b) & \text{otherwise} \end{cases}$$

- ∂ is a β rule and let F_1, F_2 be such that $F \rhd F_1, F_2$ on the respective branches B_1, B_2 with the corresponding leaves L_1, L_2. The mapping $\mu' : \pi' \to \sigma(T)$ is defined as follows for every branch b of π':

$$\mu'(b) = \begin{cases} \mu(B) \cup \{F_1\} & \text{if } b \text{ is } B_1 \\ \mu(B) \cup \{F_2\} & \text{if } b \text{ is } B_2 \\ \mu(b) & \text{otherwise} \end{cases}$$

- ∂ is a δ rule. There are two cases. If F has no dependency, then this case is the same as the α and γ rules. Otherwise, we generate an initial mapping λ between the proof tree post-weakening π'' (where ($F\rhd$) F' is in ($B \sqsubset$) B' in π'') and π that maps every branch of π'' that is not B' to itself and B' to its maximal initial segment B'' such that $\mathcal{L}(B'') \subseteq \mathcal{L}(B')$. By local extensibility of the strategy rules, we apply them inductively on the list of rules to reapply, and the last mapping yielded, λ', (i) maps all the branches that are not B' to themselves or to their original (if they are a copy) and (ii) that B' has recovered all the formulas of B by strict extension of the branch $\lambda'(B')$ and has never weakened F'. Indeed, F' has become exclusive to B', meaning that it cannot be weakened by Def. 17. Furthermore, by minimality of λ', we know that $\lambda'(B') = B$. Thus, a mapping to the tableau can be recovered: for every branch b of π',

$$\mu'(b) = \begin{cases} \mu(\lambda'(b)) & \text{if } b \neq B' \text{ (i.e., mapped to an existing copy of itself in } \pi) \\ \mu(b) \cup \{F'\} & \text{otherwise} \end{cases}$$

For α and γ rules, we trivially have that $\mu(B) \sqsubset \mu'(B')$ if $B \sqsubset B'$. Moreover, for β rules, we have B_1, B_2 such that $B \sqsubset B_1$ and $B \sqsubset B_2$ and $\mu(B) \sqsubset \mu'(B_1)$ and $\mu(B) \sqsubset \mu'(B_2)$. For δ rules, there are two cases. Either it has no dependency and then we enjoy the same property. Either it has dependencies, and then there are two cases for $B'' \neq B'$: either $B'' \in \mathrm{dom}(\mu)$ and in this case, $\mu(B'') = \mu'(B'')$, either $B'' \notin \mathrm{dom}(\mu)$ but is a copy of $B_0 \in \mathrm{dom}(\mu)$ and thus $\mu(B'') = \mu(B_0)$. Furthermore, $\mu(B) \sqsubset \mu'(B')$. In every case, we have a non-regression of the mapping and a strict progression for at least one branch. □

We then use the last condition of the local extensibility to show that there is a finite amount of branches that can be generated by the algorithm. The termination will then naturally follow of the co-well-foundedness of the order $\sqsubset$.

Lemma 20. *Given a set of locally extensible strategy rules, the strategy generates a finite amount of branches.*

Proof. Suppose that there exists a branch that is generated an infinite amount of times. It means that the under-construction GS3 proof tree has generated an infinite path (by König's lemma [34]). It also means that at least one relevant formula is missing in each node of this path. However, by local extensibility of the strategy rules, all the exclusive relevant formulas are kept while branching. In addition, by Cor. 16, we know that all the relevant formulas will be generated in the children of this branch, thus all of its children will be closed. As we are in a binary tree, we are finitely branching and thus there can be no infinite path on this branch as all the children are finite. Thus this branch generates a finite amount of subbranches. As all branches generate a finite amount of subbranches, in particular the root of the tree also generates a finite amount of branches, thus the proof has a finite amount of branches. □

Theorem 21 (Termination of the Deskolemization Strategy)**.** *If a set of strategy rules are locally extensible, then the instantiation of the deskolemization strategy with those rules terminates when translating a tableau (T, σ) to a GS3 proof tree π.*

Proof. By Lem. 20 and well-foundedness of $\sqsubset^{-1}$, all tableau rules are (finitely) integrated in a branch and the number of branches is finite. □

Theorem 22 (Soundness)**.** *Provided local extensibility of the strategy rules, any instantiation of the deskolemization strategy with those rules is a sound algorithm.*

Proof. Let (T, σ) be a closed tableau. By Thm. 21, the deskolemization strategy yields a GS3 proof tree and a mapping $\mu : \pi \to \sigma(T)$ (by Lem. 19). By μ and local extensibility of the strategy rules, π is well-formed and is a proof of the formula at the root of T as all leaves of π contain a contradiction. □

It is important to note that this soundness framework is designed to be generic, but has one lemma that depends on the tableaux rules, Lem. 19. As such, it is presented here for first-order tableaux but can be extended with rules other than the usual α, β, δ and γ (for instance, when dealing with rewriting rules) by specifying the mapping of the new rules in Lem. 19 and showing their progression.

5 Instantiation of the Strategy for First-Order Logic

In this section, we introduce two *strategy rules* to reapply the rules, that regenerate the formulas weakened during deskolemization of free-variable first-order tableaux.

Rule 1. *If the rule to reapply is a unary inference rule, then reapply it as is.*

Lemma 23. *Rule 1 is locally extensible.*

Proof (Sketch). Remark that if a δ rule should be reapplied, then the δ-term it introduces is legal. Hence, a unary rule can be seamlessly integrated in the branch B, yielding a branch B', and the mapping of B' is given by the maximum initial segment of B' such that its labelling formulas are fully contained in $\mathcal{L}(B')$. □

For β rules, the *strategy rule* is not as trivial: it is important to be careful so as to not reapply a rule that has already introduced a known Skolem symbol. We have already introduced all the necessary notions to know which formulas can be safely kept or removed.

Rule 2. *If the rule to reapply is a β rule, then reapply it and launch a subroutine that weakens all the non-exclusive formulas (except the ones that are naturally present in the branch) of a node in the non-focused branch and its (future) extensions.*

Lemma 24. *Rule 2 is locally extensible.*

Proof (Sketch). The application of a β rule on a branch B creates two branches. There is almost nothing to do on the focused branch and it straightforwardly leads to a strict extension of B. On the non-focused branch B', define π_+ by taking the subtree π rooted at the first node of the branch containing the applied β rule that is not an initial segment of B, and augment each node of π_+ by the node-exclusive formulas of $\mathcal{L}(B)$. π_+ is then grafted on the leaf of B'. This ensures the non-regression of the mapping. □

Rule 1 and Rule 2 define an instantiation of the strategy with rules locally extensible. By Thm. 22, the conjunction of the previous lemmas prove that the instantiation of the strategy with Rule 1 and Rule 2 is a sound deskolemization algorithm for first-order tableaux.

6 Implementation and Experimental Results

To obtain machine-checkable proofs, we implement a translation process in two steps: deskolemization and translation. Deskolemization transforms a tableau proof into a deskolemized GS3 proof and can be found in Goéland's public repository[1]. The advantage of this step is that it produces a sequent proof that is easily checkable by any proof assistant, as GS3 can be embedded in most such tools. For instance, two embeddings of GS3 into Coq [8] and Lambdapi [7] have been implemented in Goéland. Some similar embedding has also been implemented in Zenon [13, 20].

Moreover, Goéland features an extension to reason efficiently within theories using deduction modulo theory (DMT) [24]. As the prover uses this extension to reduce the proof size, we also consider an output for proofs with rewrite rule steps.

This section shows that our algorithm has real-world value by deskolemizing state-of-the-art proofs and that translation is affordable in practice, even though it is theoretically exponential. As the *de facto* standard of automated reasoning, we have benchmarked our approach on problems from the TPTP [39] library. The results are summarized in Table 1.

Methodology. We selected problems from two categories with FOF theorems: syntactic problems (SYN) and naive set theory problems (SET). SYN includes problems that are made mostly to test different properties of ATP systems [10]: passing these test problems gives good reasons to assume the tool should behave as expected for all the other categories. SET proposes standard reasoning over real-world problems within a theory. These problems are in general harder and DMT usually helps [16], as the axioms of set theory are good targets for rewriting. We ran experiments on a HPC platform with 28 cores (Intel Xeon E5-2680 v4 2.4 GHz), 128 gigabytes of RAM, and a 300s timeout. The sets of problems, together with Goéland and the benchmark scripts, are available online[2]. Note that, as Goéland is a parallel theorem prover,

[1] https://github.com/GoelandProver/Goeland in the folder `src/proof_output/gs3`

[2] Benchmarked problems available at https://github.com/GoelandProver/GoelandBenchmarks/ in the `PROOF_CERTIFICATION` folder.

	Problems Proved	Avg. proof size	Avg. size increase	Max. size increase	Avg. time deskolemization (ms)	Avg. time translation GS3 to Coq (ms)
Goéland	261	6.9	0 %	—	72.1	15.5
Goéland+δ^+	272	7.0	8.1 %	× 5.3	75.8	14.4
Goéland+δ^{+^+}	274	7.1	10.6 %	× 10.3	134.1	39.3
Goéland +DMT	363	6.4	0 %	—	63.4	11.1
Goéland +DMT+δ^+	375	6.5	4.5 %	× 3.9	72.1	12.1
Goéland +DMT+δ^{+^+}	377	6.5	7.4 %	× 5.2	76.1	12.1

Table 1: Comparison between the different Skolemization strategies, their proof-size increase and the deskolemization and translation times (in ms).

its proof-search algorithm is *non-deterministic* and as such, results presented here may not be perfectly reproducible. The standard version of Goéland performs *outer* Skolemization and is used as a baseline for comparison with more advanced Skolemization techniques. Each variants comes with its corresponding instantiation of the deskolemization algorithm. We evaluated the six following variants: Goéland, Goéland+δ^+, Goéland+δ^{+^+}, Goéland+DMT, Goéland+DMT+δ^+ and Goéland+DMT+δ^{+^+}. Each variant corresponds to a particular option set of Goéland, where the δ^+ rules can be activated using the `-inner` flag, δ^{+^+} rules with the `-preinner` flag and DMT with the `-dmt` flag.

Insight into the Results. Table 1 presents an overview of the results for the above benchmark. The result of each variant is displayed in a row, that successively contains: the number of problems on which a *tableau proof* has been output by the variant ; the average size of the tableau proof in terms of number of branches ; the average percentage and maximal ratio size increase between the tableau and the Coq proofs, also in terms of the number of branches ; the average translation times from tableaux to GS3 (deskolemization) and from GS3 to Coq (embedding), both in milliseconds. All the problems proved by Goéland have been successfully translated to Coq.

The results obtained are very promising as (i) every proof of every variant is properly certified and (ii) the average size increase between the two versions of the proof is low. In theory, a tableau proof can be at least exponentially better than its GS3 counterpart in inner Skolemization which in turn is exponentially better than for pre-inner Skolemization. However, we notice that for both variants featuring these Skolemization strategies, the average and maximum increase of size is low (far away from the bound), while the DMT variant is even better.

7 Conclusion

In summary, we proposed a new generic strategy to deskolemize tableaux proofs with different Skolemization strategies by combining ideas from [14, 28], implemented a soundness framework

for this strategy, instantiated it for first-order tableaux and showed the soundness of the approach. This strategy shows promises: as far as we know, it is the first one that generalizes properly to more optimized Skolemization strategies. As such, the authors hope that it can become a basis for further deskolemization investigations.

In parallel, we implemented this strategy in Goéland, a tool that provides a standard tableaux-proof output, for both δ^+- and δ^{++}- rules. This enabled the translation of its proofs to GS3. Then, we built a Coq and Lambdapi embedding of GS3: this allowed an implementation, in Goéland, of a translation from GS3 to Coq and Lambdapi. All in all, the results obtained are satisfactory as they validate the translation algorithm by yielding relatively short proofs. It means that it is, in practice, affordable to certify proofs using the proposed translation algorithm together with an embedding in a proof assistant.

With our approach validated, our next goal is to write an independent tool that understands a language which ATP can easily output (say, TSTP [38]) and implements different deskolemization strategies, both for (clausal and non-clausal) tableaux and non-tableaux provers. In parallel, we want to develop the idea around the creation of lemmas [26, 32] and see its impact on the theoretical bound. We also aim at extending our framework, by investigating how to instantiate the strategy in the context of deduction modulo theory or other logics.

Finally, we also want to broaden the scope of this method by providing translations to other proof assistants, i.e., Agda [15], Isabelle/HOL [36], Lean [22], Lisa [30] or the Mizar Project [6], potentially using existing approaches for proof translation (in particular, those developed around Dedukti/Lambdapi).

References

[1] P. B. Andrews. Theorem proving via general matings. *Journal of the ACM (JACM)*, 28(2):193–214, 1981.

[2] A. Assaf, G. Burel, R. Cauderlier, D. Delahaye, G. Dowek, C. Dubois, F. Gilbert, P. Halmagrand, O. Hermant, and R. Saillard. Dedukti: a logical framework based on the $\lambda\pi$-calculus modulo theory. 2016.

[3] J. Avigad. Eliminating Definitions and Skolem Functions in First-Order Logic. In *16th Annual IEEE Symposium on Logic in Computer Science*, pages 139–146. IEEE Computer Society, 2001.

[4] M. Baaz and C. G. Fermüller. Non-elementary Speedups between Different Versions of Tableaux. In P. Baumgartner, R. Hähnle, and J. Posegga, editors, *TABLEAUX '95*, volume 918 of *Lecture Notes in Computer Science*, pages 217–230. Springer, 1995.

[5] M. Baaz, S. Hetzl, and D. Weller. On the Complexity of Proof Deskolemization. *J. Symb. Log.*, 77(2):669–686, 2012.

[6] G. Bancerek, C. Byliński, A. Grabowski, A. Korniłowicz, R. Matuszewski, A. Naumowicz, K. Pak, and J. Urban. Mizar: State-of-the-art and beyond. In *International Conference on Intelligent Computer Mathematics*, pages 261–279. Springer, 2015.

[7] H. P. Barendregt, W. Dekkers, and R. Statman. *Lambda calculus with types*. Cambridge University Press, 2013.

[8] B. Barras, S. Boutin, C. Cornes, J. Courant, J.-C. Filliatre, E. Gimenez, H. Herbelin, G. Huet, C. Munoz, C. Murthy, et al. *The Coq proof assistant reference manual: Version 6.1*. PhD thesis, Inria, 1997.

[9] B. Beckert, R. Hähnle, and P. H. Schmitt. The even more liberalized δ-rule in free variable semantic tableaux. In G. Gottlob, A. Leitsch, and D. Mundici, editors, *Computational Logic and Proof Theory*, pages 108–119, Berlin, Heidelberg, 1993. Springer Berlin Heidelberg.

[10] C. Benzmüller and C. E. Brown. A structured set of higher-order problems. In *International Conference on Theorem Proving in Higher Order Logics*, 2005.

[11] E. W. Beth. *Formal Methods: An Introduction to Symbolic Logic and to the Study of Effective Operations in Arithmetic and Logic*, volume 4 of *Synthese Library*. D. Reidel Pub. Co., 1962.

[12] W. Bibel. *Automated theorem proving*. Vieweg, 1982.

[13] R. Bonichon, D. Delahaye, and D. Doligez. Zenon: An extensible automated theorem prover producing checkable proofs. In *Logic for Programming, Artificial Intelligence and Reasoning*, pages 151–165. Springer, 2007.

[14] R. Bonichon and O. Hermant. A Syntactic Soundness Proof for Free-Variable Tableaux with on-the-fly Skolemization. 2013.

[15] A. Bove, P. Dybjer, and U. Norell. A Brief Overview of Agda – A Functional Language with Dependent Types. In Berghofer, Stefan and Nipkow, Tobias and Urban, Christian and Wenzel, Makarius, editor, *Theorem Proving in Higher Order Logics*, pages 73–78, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[16] G. Burel, G. Bury, R. Cauderlier, D. Delahaye, P. Halmagrand, and O. Hermant. First-order automated reasoning with theories: When deduction modulo theory meets practice. *Journal of Automated Reasoning*, 64(6):1001–1050, 2020.

[17] J. Cailler, J. Rosain, D. Delahaye, S. Robillard, and H. L. Bouziane. Goéland: a concurrent tableau-based theorem prover (system description). In *IJCAR 2022-11th International Joint Conference on Automated Reasoning*, volume 13385, pages 359–368, 2022.

[18] D. Cantone and M. N. Asmundo. A Further and Effective Liberalization of the δ-Rule in Free Variable Semantic Tableaux. In R. Caferra and G. Salzer, editors, *Automated Deduction in Classical and Non-Classical Logics, Selected Papers*, volume 1761 of *Lecture Notes in Computer Science*, pages 109–125. Springer, 1998.

[19] D. Cantone and M. Nicolosi-Asmundo. A sound framework for δ-rule variants in free-variable semantic tableaux. *Journal of Automated Reasoning*, 38:31–56, 2007.

[20] R. Cauderlier and P. Halmagrand. Checking zenon modulo proofs in dedukti. 2015.

[21] K. Chaudhuri, M. Manighetti, and D. Miller. A Proof-Theoretic Approach to Certifying Skolemization. In A. Mahboubi and M. O. Myreen, editors, *CPP 2019*, pages 78–90. ACM, 2019.

[22] L. M. de Moura, S. Kong, J. Avigad, F. van Doorn, and J. von Raumer. The Lean Theorem Prover (System Description). In A. P. Felty and A. Middeldorp, editors, *CADE-25*, volume 9195 of *Lecture Notes in Computer Science*, pages 378–388. Springer, 2015.

[23] N. Dershowitz and Z. Manna. Proving termination with multiset orderings. *Commun. ACM*, 22:465476, 1979.

[24] G. Dowek, T. Hardin, and C. Kirchner. Theorem Proving Modulo. *Journal of Automated Reasoning (JAR)*, 31(1):33–72, 2003.

[25] G. Dowek and B. Werner. A constructive proof of skolem theorem for constructive logic. 2005.

[26] G. Ebner, S. Hetzl, A. Leitsch, G. Reis, and D. Weller. On the generation of quantified lemmas. *Journal of Automated Reasoning*, 63:95–126, 2019.

[27] M. Färber and C. Kaliszyk. No Choice: Reconstruction of First-order ATP Proofs without Skolem Functions. In P. Fontaine, S. Schulz, and J. Urban, editors, *Proceedings of the 5th Workshop on Practical Aspects of Automated Reasoning*, volume 1635 of *CEUR Workshop Proceedings*, pages 24–31. CEUR-WS.org, 2016.

[28] M. Fitting. *First-Order Logic and Automated Theorem Proving*. Springer, 1990.

[29] M. Giese and W. Ahrendt. Hilbert's ϵ-terms in automated theorem proving. In N. V. Murray, editor, *Automated Reasoning with Analytic Tableaux and Related Methods*, pages 171–185, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg.

[30] S. Guilloud, S. Gambhir, and V. Kuncak. LISA – A Modern Proof System. In *14th Conference on Interactive Theorem Proving*, Leibniz International Proceedings in Informatics, pages 17:1–17:19, Bialystok, 2023. Daghstuhl.

[31] R. Hähnle and P. H. Schmitt. The Liberalized δ-Rule in Free Variable Semantic Tableaux. *J.*

Autom. Reason., 13(2):211–221, 1994.

[32] S. Hetzl, A. Leitsch, and D. Weller. Towards algorithmic cut-introduction. In *Logic for Programming, Artificial Intelligence, and Reasoning: 18th International Conference, LPAR-18, Mérida, Venezuela, March 11-15, 2012. Proceedings 18*, pages 228–242. Springer, 2012.

[33] J. Hintikka. Two Papers on Symbolic Logic: Form and Content in Quantification Theory and Reductions in the Theory of Types. *Societas Philosophica, Acta philosophica Fennica*, 8:7–55, 1955.

[34] D. Knig. Über eine schlussweise aus dem endlichen ins unendliche. 1927.

[35] D. A. Miller. A Compact Representation of Proofs. *Stud Logica*, 46(4):347–370, 1987.

[36] T. Nipkow, M. Wenzel, and L. C. Paulson. *Isabelle/HOL: a proof assistant for higher-order logic*. Springer, 2002.

[37] A. Schlichtkrull, J. C. Blanchette, and D. Traytel. A verified prover based on ordered resolution. In A. Mahboubi and M. O. Myreen, editors, *Proceedings of the 8th ACM SIGPLAN International Conference on Certified Programs and Proofs, CPP 2019, Cascais, Portugal, January 14-15, 2019*, pages 152–165. ACM, 2019.

[38] G. Sutcliffe. TPTP, TSTP, CASC, etc. In V. Diekert, M. Volkov, and A. Voronkov, editors, *Proceedings of the 2nd International Computer Science Symposium in Russia*, number 4649 in Lecture Notes in Computer Science, pages 7–23. Springer-Verlag, 2007.

[39] G. Sutcliffe. The TPTP Problem Library and Associated Infrastructure. From CNF to TH0, TPTP v6.4.0. *Journal of Automated Reasoning (JAR)*, 59(4):483–502, 2017.

[40] A. S. Troelstra and H. Schwichtenberg. *Basic proof theory, Second Edition*, volume 43 of *Cambridge tracts in theoretical computer science*. Cambridge University Press, 2000.

[41] D. van Dalen. *Logic and structure (5. ed.)*. Universitext. Springer, 2013.