TableauxRocq: A Deep-Embedding of Free-Variable Tableaux in Rocq

ITP 2026, Lisbon, Portugal

Johann Rosain[1] Julie Cailler[2]

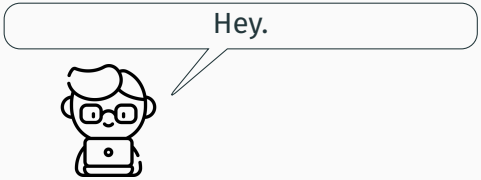
July 28, 2026

[1]École Normale Supérieure de Lyon, Lyon, France

[2]University of Lorraine, CNRS, Inria, LORIA, Nancy, France

Once Upon a Time...

Once Upon a Time...



Once Upon a Time...

I've been meaning to mecanize our paper.

Hey.

Once Upon a Time...

Do you know about some library for free-variable tableaux in a proof assistant?

I've been meaning to mecanize our paper.

Hey.

Once Upon a Time...

Do you know about some library for free-variable tableaux in a proof assistant?

I've been meaning to mecanize our paper.

Hey.

Hi.

Once Upon a Time...

Do you know about some library for free-variable tableaux in a proof assistant?

I've been meaning to mecanize our paper.

Hey.

I don't.

Hi.

Once Upon a Time...

Do you know about some library for free-variable tableaux in a proof assistant?

I've been meaning to mecanize our paper.

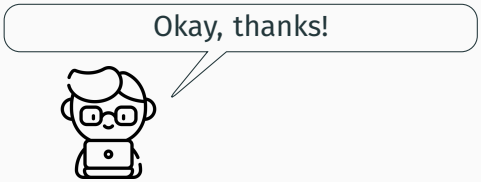
Hey.

I've also asked around and it doesn't seem like there is one.

I don't.

Hi.

Once Upon a Time...



Once Upon a Time...

I'll make one, then.

Okay, thanks!

Once Upon a Time...

It'll be small though, with only have some easy results, like soundness.

I'll make one, then.

Okay, thanks!

Once Upon a Time...

It'll be small though, with only have some easy results, like soundness.

I'll make one, then.

Okay, thanks!

Sure. That's already a good first step.

Once Upon a Time...

It'll be small though, with only have some easy results, like soundness.

I'll make one, then.

Okay, thanks!

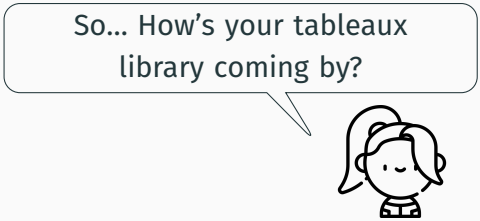
Good luck!

Sure. That's already a good first step.

Once Upon a Time...

Some months later...

Once Upon a Time...

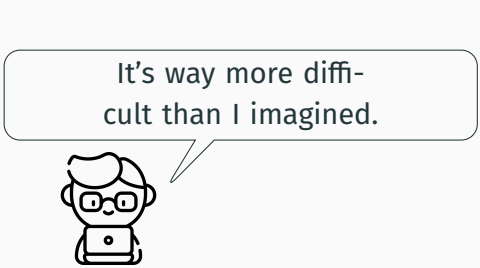


Once Upon a Time...

It's way more difficult than I imagined.

So... How's your tableaux library coming by?

Once Upon a Time...



Once Upon a Time...

Do you have 20 minutes?

It's way more difficult than I imagined.

Oh! Why?

So... How's your tableaux library coming by?

Once Upon a Time...

Do you have 20 minutes?

It's way more difficult than I imagined.

Sure.

Oh! Why?

So... How's your tableaux library coming by?

Once Upon a Time...

Then let me tell you about it.

Do you have 20 minutes?

It's way more diffi-
cult than I imagined.

Sure.

Oh! Why?

So... How's your tableaux
library coming by?

FOL, But Not Fool

Small locally nameless syntax:

$$
\begin{array}{llll}
t, t_i & ::= & n \in \mathbb{N} \mid X \in \mathcal{X} \mid f(t_1, \ldots, t_n) & f \in \mathcal{F} \\
F, G & ::= & \bot \mid P(t_1, \ldots, t_n) \mid \neg F \mid F \vee G \mid \forall F & P \in \mathcal{P}
\end{array}
$$

with operations:

- instantiation $F[t]$,
- substitution $F\sigma$, $\sigma = \{X_1 \mapsto t_1, \ldots, X_n \mapsto t_n\}$.

FOL, But Not Fool

Small locally nameless syntax:

$$\begin{array}{llll} t, t_i & ::= & n \in \mathbb{N} \mid X \in \mathcal{X} \mid f(t_1, \ldots, t_n) & f \in \mathcal{F} \\ F, G & ::= & \bot \mid P(t_1, \ldots, t_n) \mid \neg F \mid F \vee G \mid \forall F & P \in \mathcal{P} \end{array}$$

with operations:

- instantiation $F[t]$,
- substitution $F\sigma$, $\sigma = \{X_1 \mapsto t_1, \ldots, X_n \mapsto t_n\}$.

Free-variable tableaux, *en bref*:

- proofs by contradiction,
- one-sided upside-down sequents,
- placeholder (in $\mathcal{X}$) awaiting term instantiation (given by substitution).

Elementary, Dear Audience

Axioms (i.e., contradictions):

$$\frac{\bot}{\odot}\odot_{\bot} \qquad\qquad \frac{F, \neg G}{\sigma}\odot_{\sigma}, F\sigma = G\sigma$$

Elementary, Dear Audience

Axioms (i.e., contradictions):

$$\frac{\bot}{\odot}\odot_{\bot} \qquad\qquad \frac{F, \neg G}{\sigma}\odot_{\sigma}, F\sigma = G\sigma$$

Expansion:

$$\frac{\neg\neg F}{F}\alpha_{\neg\neg} \qquad \frac{\neg(F \vee G)}{\neg F, \neg G}\alpha_{\neg\vee} \qquad \frac{F \vee G}{F \quad G}\beta_{\vee}$$

Elementary, Dear Audience

Axioms (i.e., contradictions):

$$\frac{\bot}{\odot}\odot_{\bot} \qquad\qquad \frac{F, \neg G}{\sigma}\odot_{\sigma}, F\sigma = G\sigma$$

Expansion:

$$\frac{\neg\neg F}{F}\alpha_{\neg\neg} \qquad\qquad \frac{\neg(F \vee G)}{\neg F, \neg G}\alpha_{\neg\vee} \qquad\qquad \frac{F \vee G}{F \quad G}\beta_{\vee}$$

Instantiation:

$$\frac{\forall F}{F[X]}\gamma_{\forall} \qquad\qquad \frac{\neg(\forall F)}{\neg F[\mathrm{sko}(X_1, \ldots, X_n)]}\delta_{\neg\forall}$$

Wait, it's All Choice? Always Has Been

$\mathrm{sko}(X_1, \ldots, X_n)$?

Wait, it's All Choice? Always Has Been

$\mathrm{sko}(X_1, \ldots, X_n)? \to$ "suitable" term.

Wait, it's All Choice? Always Has Been

$\mathrm{sko}(X_1, \ldots, X_n)? \to$ "suitable" term. Multiple suitable terms, e.g.:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\forall\neg(\neg D(0) \vee \forall D(0))}{\neg(\neg D(X) \vee \forall D(0))}\,\gamma_\forall}{\neg\neg D(X), \neg(\forall D(0))}\,\alpha_{\neg\vee}}{D(X), \neg(\forall D(0))}\,\alpha_{\neg\neg}}{\neg D(f(X)), \forall\neg(\neg D(0) \vee \forall D(0))}\,\delta_{\neg\forall}}{\neg(\neg D(Y) \vee \forall D(0))}\,\gamma_\forall}{\neg\neg D(Y), \neg(\forall D(0))}\,\alpha_{\neg\vee}}{D(Y), \neg D(f(X))}\,\alpha_{\neg\neg}}{\{Y \mapsto f(X)\}}\,\odot$$

"Outer" Skolemization

Wait, it's All Choice? Always Has Been

sko$(X_1, \ldots, X_n)$? $\rightarrow$ "suitable" term. Multiple suitable terms, e.g.:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\forall\neg(\neg D(0) \vee \forall D(0))}{\neg(\neg D(X) \vee \forall D(0))}\,\gamma_\forall}{\neg\neg D(X), \neg(\forall D(0))}\,\alpha_{\neg\vee}}{D(X), \neg(\forall D(0))}\,\alpha_{\neg\neg}}{\neg D(f(X)), \forall\neg(\neg D(0) \vee \forall D(0))}\,\delta_{\neg\forall}}{\neg(\neg D(Y) \vee \forall D(0))}\,\gamma_\forall}{\neg\neg D(Y), \neg(\forall D(0))}\,\alpha_{\neg\vee}}{D(Y), \neg D(f(X))}\,\alpha_{\neg\neg}}{\{Y \mapsto f(X)\}}\,\odot$$

"Outer" Skolemization

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\forall\neg(\neg D(0) \vee \forall D(0))}{\neg(\neg D(X) \vee \forall D(0))}\,\gamma_\forall}{\neg\neg D(X), \neg(\forall D(0))}\,\alpha_{\neg\vee}}{D(X), \neg(\forall D(0))}\,\alpha_{\neg\neg}}{\neg D(c), D(X)}\,\delta_{\neg\forall}}{\{X \mapsto c\}}\,\odot$$

"Inner" Skolemization

Is This a Sound Proof System?

Many different Skolemizations. Goals:

- smaller proof-search space so faster automated provers,
- more compact proofs,
- retain *semantic* soundness.

Is This a Sound Proof System?

Many different Skolemizations. Goals:

- smaller proof-search space so faster automated provers,
- more compact proofs,
- retain *semantic* soundness.

Issue: most Skolemizations are **not** *syntactically* sound! (e.g., inner).

Is This a Sound Proof System?

Many different Skolemizations. Goals:

- smaller proof-search space so faster automated provers,
- more compact proofs,
- retain *semantic* soundness.

Issue: most Skolemizations are **not** *syntactically* sound! (e.g., inner).

How do we justify these unsound systems?

- Syntactic soundness via translation.[1]
- Semantic soundness via deep embedding (this work).

[1] *A Generic Deskolemization Strategy*, Rosain et al. (LPAR 2024).

Mom, Can We Have a Proof Certificate at Home?

Side-bonus of this approach: nicer proof certificates.

Mom, Can We Have a Proof Certificate at Home?

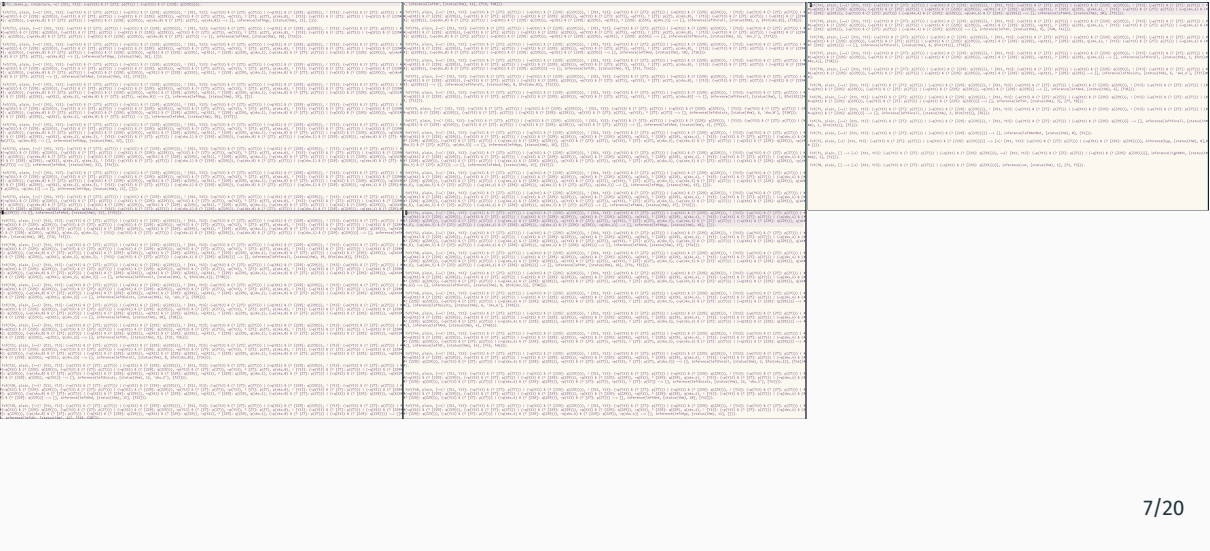
Side-bonus of this approach: nicer proof certificates.

Consider:

$$\dfrac{\dfrac{\dfrac{\forall\forall((\neg P(0) \wedge \exists P(0)) \vee (\neg Q(1) \wedge \exists Q(0)))}{(\neg P(Y) \wedge \exists P(0)) \vee (\neg Q(X) \wedge \exists Q(0))}\,\gamma_\forall \times 2}{\neg P(Y) \wedge \exists P(0)}\,\beta_\vee \qquad \dfrac{\neg Q(X) \wedge \exists Q(0)}{\neg Q(X), \exists Q(0)}\,\alpha_\wedge}{\dfrac{\neg P(Y), \exists P(0)}{P(c), \neg P(Y)}\,\delta_\exists \qquad \dfrac{\neg Q(X), \exists Q(0)}{Q(d), \neg Q(X)}\,\delta_\exists}$$
$$\dfrac{P(c), \neg P(Y)}{\{Y \mapsto c\}}\,\odot \qquad \dfrac{Q(d), \neg Q(X)}{\{X \mapsto d\}}\,\odot$$

Does not look particularly bad. Right...?

(Mom) No, There Is a Proof Certificate at Home:



(Mom) No, There Is a Proof Certificate at Home:

Over 100 lines of 500+ characters proof-steps from a 5-lines proof!!!

You Know, I'm Somewhat of a Proof Certificate Myself

In our system:[1]

```
fof(goal, definition,
    ! [X, Y] : (~p(Y) & (? [Z]: p(Z))) | (~q(X) & (? [Z] : q(Z)))).
fof(desko, negated_conjecture, goal).
fof(s, substitution, { X -> d ; Y -> c }, inner).

fof(s0, plain, goal, inference(forall, [s1], fot(X)).
fof(s1, plain, ! [Y] : (~p(Y) & (? [Z]: p(Z))) | (~q(X) & (? [Z] : q(Z))),
    inference(forall, [s2], fot(Y)).
fof(s2, plain, (~p(Y) & (? [Z]: p(Z))) | (~q(X) & (? [Z] : q(Z))), inference(or, [
    s3, s4])).
fof(s3, plain, ~p(Y) & (? [Z]: p(Z)), inference(and, [s5])).
fof(s5, plain, ? [Z]: p(Z), inference(exists, [s6], fot(c)).
fof(s6, plain, [~p(Y), p(c)], inference(hyp, []).
fof(s4, plain, ~q(X) & (? [Z]: q(Z)), inference(and, [s7])).
fof(s7, plain, ? [Z]: q(Z), inference(exists, [s8], fot(d)).
fof(s8, plain, [~q(X), q(d)], inference(hyp, []).
```

[1] Non-contractual format. Subject to change.

You Know, I'm Somewhat of a Proof Certificate Myself

In our system:[1]

```
fof(goal, definition,
    ! [X, Y] : (~p(Y) & (? [Z]: p(Z))) | (~q(X) & (? [Z] : q(Z)))).
fof(desko, negated_conjecture, goal).
fof(s, substitution, { X -> d ; Y -> c }, inner).
```

How?

```
fof(s2, plain, (~p(Y) & (? [Z]: p(Z))) | (~q(X) & (? [Z] : q(Z))), inference(or, [
    s3, s4])).
fof(s3, plain, ~p(Y) & (? [Z]: p(Z)), inference(and, [s5])).
fof(s5, plain, ? [Z]: p(Z), inference(exists, [s6], fot(c)).
fof(s6, plain, [~p(Y), p(c)], inference(hyp, [])).
fof(s4, plain, ~q(X) & (? [Z]: q(Z)), inference(and, [s7])).
fof(s7, plain, ? [Z]: q(Z), inference(exists, [s8], fot(d)).
fof(s8, plain, [~q(X), q(d)], inference(hyp, [])).
```

[1] Non-contractual format. Subject to change.

Trade Offer—I Receive: a Proof, You Receive: Validity

An object $\langle \Gamma, \pi \rangle : \mathsf{Proof}$ is:

- a list of formulas, and
- a binary tree of rules.

Trade Offer—I Receive: a Proof, You Receive: Validity

An object $\langle \Gamma, \pi \rangle : \mathsf{Proof}$ is:

- a list of formulas, and
- a binary tree of rules.

Predicate $\mathsf{isTableau} : \mathsf{Proof} \to \mathsf{Sub} \to \mathsf{Prop}$, want:

Soundness

For every $\langle \Gamma \blacktriangleright \neg F, \pi \rangle : \mathsf{Proof}$ and $\sigma : \mathsf{Sub}$,

$$\mathsf{isTableau}\ \langle \Gamma \blacktriangleright \neg F, \pi \rangle\ \sigma \implies \Gamma \models F.$$

Here, we work with Tarski's semantics.

Why Is it, When Something Happens, it Is Always You?

Tableaux: one-sided sequents. Usually formalized as inductive predicate.

- Induction builds the “proof tree”.
- Constructor = rule.

Why Is it, When Something Happens, it Is Always You?

Tableaux: one-sided sequents. Usually formalized as inductive predicate.

- Induction builds the “proof tree”.
- Constructor = rule.

Issue

Syntactically unsound:

- $M(\Gamma) \wedge M(\neg(F[\mathrm{sko}(X_1, \ldots, X_n)]))$ proves false.
- Want: $M(\Gamma)$ proves false, with $\neg(\forall F) \in \Gamma$. Assume $M(\Gamma)$.
- Show: $M(\Gamma), M(\neg(F[\mathrm{sko}(X_1, \ldots, X_n)]))$.

But $M(\neg(\forall F))$ **does not** entail $M(\neg(F[\mathrm{sko}(X_1, \ldots, X_n)]))$.

Why Is it, When Something Happens, it Is Always You?

Tableaux: one-sided sequents. Usually formalized as inductive predicate.

- Induction builds the “proof tree”.
- Constructor = rule.

Issue

Syntactically unsound:

- $M(\Gamma) \wedge M(\neg(F[\mathrm{sko}(X_1, \ldots, X_n)]))$ proves false.
- Want: $M(\Gamma)$ proves false, with $\neg(\forall F) \in \Gamma$. Assume $M(\Gamma)$.
- Show: $M(\Gamma), M(\neg(F[\mathrm{sko}(X_1, \ldots, X_n)]))$.

But $M(\neg(\forall F))$ **does not** entail $M(\neg(F[\mathrm{sko}(X_1, \ldots, X_n)]))$.

Direct proof fails. Other proof techniques?

You Fool! I Have 70 Alternative Proofs!

What does literature[2,3,4] says?

[2] *First-order Logic and Automated Theorem Proving*, Fitting (1996)
[3] *The Liberalized δ-Rule in Free Variable Semantic Tableaux*, Hähnle and Schmitt (1994)
[4] *The even more liberalized δ-rule in free variable semantic tableaux*, Beckert et al. (1993)

You Fool! I Have 70 Alternative Proofs!

What does literature[2,3,4] says?

Satisfiable context implies satisfiable tableau.

[2] *First-order Logic and Automated Theorem Proving*, Fitting (1996)
[3] *The Liberalized δ-Rule in Free Variable Semantic Tableaux*, Hähnle and Schmitt (1994)
[4] *The even more liberalized δ-rule in free variable semantic tableaux*, Beckert et al. (1993)

You Fool! I Have 70 Alternative Proofs!

What does literature[2,3,4] says?

Satisfiable context implies satisfiable tableau.

Two approaches:

- build it iteratively, or
- build it *before* inspecting the proof.

[2] *First-order Logic and Automated Theorem Proving*, Fitting (1996)
[3] *The Liberalized δ-Rule in Free Variable Semantic Tableaux*, Hähnle and Schmitt (1994)
[4] *The even more liberalized δ-rule in free variable semantic tableaux*, Beckert et al. (1993)

You Fool! I Have 70 Alternative Proofs!

What does literature[2,3,4] says?

Satisfiable context implies satisfiable tableau.

Two approaches:

- build it iteratively, or
- build it *before* inspecting the proof.

Second approach: difficult to mechanize. We go with the first one.

[2] *First-order Logic and Automated Theorem Proving*, Fitting (1996)
[3] *The Liberalized δ-Rule in Free Variable Semantic Tableaux*, Hähnle and Schmitt (1994)
[4] *The even more liberalized δ-rule in free variable semantic tableaux*, Beckert et al. (1993)

You Fool! I Have 70 Alternative Proofs!

What does literature[2,3,4] says?

Satisfiable context implies satisfiable tableau.

Two approaches:

- build it iteratively, or
- build it *before* inspecting the proof.

Second approach: difficult to mechanize. We go with the first one.

But with inductive definition, β rule fails: 2 induction hypotheses.

[2] *First-order Logic and Automated Theorem Proving*, Fitting (1996)
[3] *The Liberalized δ-Rule in Free Variable Semantic Tableaux*, Hähnle and Schmitt (1994)
[4] *The even more liberalized δ-rule in free variable semantic tableaux*, Beckert et al. (1993)

Friendship Ended with Inductives. Now Binary Relations Are My Best Friends.

Insight: cannot consider tableaux *locally*, only *globally*.

$\rightarrow$ inductive predicate not adapted here!

Insight: cannot consider tableaux *locally*, only *globally*.

$\rightarrow$ inductive predicate not adapted here!

Keep global information by tracing successive proof trees.

Insight: cannot consider tableaux *locally*, only *globally*.

$\rightarrow$ inductive predicate not adapted here!

Keep global information by tracing successive proof trees.

Tableau proof (T, σ) is:

- proof trees *sequence* s s.t. $s[i] \triangleright s[i+1]$,
- s.t. branches of *last* tree of s contain contradiction w.r.t. σ.

Insight: cannot consider tableaux *locally*, only *globally*.

$\rightarrow$ inductive predicate not adapted here!

Keep global information by tracing successive proof trees.

Tableau proof (T, σ) is:

- proof trees *sequence* s s.t. $s[i] \triangleright s[i+1]$,
- s.t. branches of *last* tree of s contain contradiction w.r.t. σ.

Note

Definition is generic w.r.t. any *semantically sound* Skolemization.

Satisfied? How Would You Rate Your Experience?

Lemma

If $T_1 \triangleright T_2$ and T_1 satisfiable, then so is T_2.

Proof.

β rule case: OK — a single induction hypothesis.

δ rule case: needs new model M' w.r.t. selected Skolemization.
Given by "soundness": $M(\neg(\forall F)) \implies M'(\neg F[t])$, t is skolemized term. □

Satisfied? How Would You Rate Your Experience?

Lemma

If $T_1 \triangleright T_2$ and T_1 satisfiable, then so is T_2.

Proof.

β rule case: OK — a single induction hypothesis.

δ rule case: needs new model M' w.r.t. selected Skolemization.
Given by "soundness": $M(\neg(\forall F)) \implies M'(\neg F[t])$, t is skolemized term. □

Actually: $\exists$ model satisfying all proof trees in sequence.

$\implies$ We believe approach of first building a model then inspecting the proof is subsumed by our method.

Satisfied? How Would You Rate Your Experience?

Lemma

If $T_1 \triangleright T_2$ and T_1 satisfiable, then so is T_2.

Proof.

β rule case: OK — a single induction hypothesis.

δ rule case: needs new model M' w.r.t. selected Skolemization.
Given by "soundness": $M(\neg(\forall F)) \implies M'(\neg F[t])$, t is skolemized term. □

Actually: $\exists$ model satisfying all proof trees in sequence.

$\implies$ We believe approach of first building a model then inspecting the proof is subsumed by our method.

Soundness follows as T_n is not satisfiable.

They Are the Same Picture

Final goal: proof-checking algorithm.

- Implemented as recursive algorithm.
- Caching of the function symbols.

They Are the Same Picture

Final goal: proof-checking algorithm.

- Implemented as recursive algorithm.
- Caching of the function symbols.

Need to show: if algorithm returns true, then $\exists$ a tableau proof. Strategy:

- Build list s of proof trees from the algorithm.
- Prove that caching at step n contains all symbols appearing in $s[n]$.
- Demonstrate that $s[i] \triangleright s[i+1]$ for every i.
- Show that $s[\text{length}(s) - 1]$ has a contradiction in every branch.

They Are the Same Picture

Final goal: proof-checking algorithm.

- Implemented as recursive algorithm.
- Caching of the function symbols.

Need to show: if algorithm returns true, then $\exists$ a tableau proof. Strategy:

- Build list s of proof trees from the algorithm.
- Prove that caching at step n contains all symbols appearing in $s[n]$.
- Demonstrate that $s[i] \triangleright s[i+1]$ for every i.
- Show that $s[\text{length}(s) - 1]$ has a contradiction in every branch.

Pretty painful but good API + automation (via tactics) helps a lot.

They Are the Same Picture

Final goal: proof-checking algorithm.

- Implemented as recursive algorithm.
- Caching of the function symbols.

Need to show: if algorithm returns true, then $\exists$ a tableau proof. Strategy:

- Build list s of proof trees from the algorithm.
- Prove that caching at step n contains all symbols appearing in $s[n]$.
- Demonstrate that $s[i] \triangleright s[i+1]$ for every i.
- Show that $s[\text{length}(s) - 1]$ has a contradiction in every branch.

Pretty painful but good API + automation (via tactics) helps a lot.

...But having inductive representation would be useful.

Is the Inductive in the Room With Us Right Now?

Is all hope lost for an inductive definition? No!

Is the Inductive in the Room With Us Right Now?

Is all hope lost for an inductive definition? No!

Proof-checking algorithm: implemented as a recursive function shown sound.

Is the Inductive in the Room With Us Right Now?

Is all hope lost for an inductive definition? No!

Proof-checking algorithm: implemented as a recursive function shown sound.

$\Longrightarrow$ can prove inductive specification of the algorithm sound.

Is the Inductive in the Room With Us Right Now?

Is all hope lost for an inductive definition? No!

Proof-checking algorithm: implemented as a recursive function shown sound.

$\Longrightarrow$ can prove inductive specification of the algorithm sound.

Interesting to maintain inductive specification internally.

- TableauxRocq: built as a library—offers more familiar API.
- More possibilities: can be easier to prove some properties on one version.
- Well-maintained API and tactics for both versions.

No Choice Needed, We Are Better

Implemented an output in TableauxRocq format in Goéland.[5]

[5] *Goéland: A Concurrent Tableau-Based Theorem Prover*, Cailler et al. (IJCAR 2022)

No Choice Needed, We Are Better

Implemented an output in TableauxRocq format in Goéland.[5]

Experiments on Skolemization-heavy family of formulas:

$$\Phi_n \triangleq \forall X_1, \ldots, X_n\ , \bigvee_{i=1}^{n} (\neg P_i(X_i) \wedge \exists Y_i,\ P_i(Y_i))$$

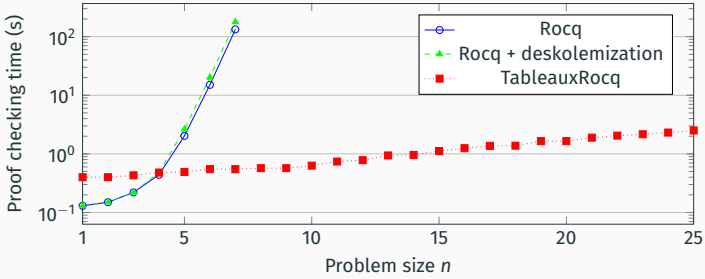
[5] *Goéland: A Concurrent Tableau-Based Theorem Prover*, Cailler et al. (IJCAR 2022)

No Choice Needed, We Are Better

Implemented an output in TableauxRocq format in Goéland.[5]

Experiments on Skolemization-heavy family of formulas:

$$\Phi_n \triangleq \forall X_1, \ldots, X_n\ ,\ \bigvee_{i=1}^{n} (\neg P_i(X_i) \wedge \exists Y_i,\ P_i(Y_i))$$



[5] *Goéland: A Concurrent Tableau-Based Theorem Prover*, Cailler et al. (IJCAR 2022)

Real-World Is Overrated, Change my Mind

Experiments (using the TPTP library):

- filter: only first-order problems without equality,
- proof search: list the problems that Goéland passes,
- proof generation: generate syntactic and semantic proof for each problem.

Real-World Is Overrated, Change my Mind

Experiments (using the TPTP library):

- filter: only first-order problems without equality,
- proof search: list the problems that Goéland passes,
- proof generation: generate syntactic and semantic proof for each problem.

Results:[2]

Output mode	Skolemization	#problems	Med. time	Avg. time	#faster
Syntactic	Outer	172	0.24s	1.74s	169
TableauxRocq			0.72s	1.91s	3
Syntactic	Inner	177	0.24s	1.70s	173
TableauxRocq			0.71s	1.38s	4

[2] Slightly differ from the ones in the paper as we have rerun them.

0 Days Since Last Lesson Learned

Takeaway of the benches:

Advantages	Inconvenients
Outputing proofs is easy Faster on proofs that use Skolemization Better scaling (faster on bigger proofs)	Slower in the given problem set

0 Days Since Last Lesson Learned

Takeaway of the benches:

Advantages	Inconvenients
Outputing proofs is easy Faster on proofs that use Skolemization Better scaling (faster on bigger proofs)	Slower in the given problem set

Instead of running it inside Rocq, extract the checker!

- Even easier to output proof (e.g., format shown in slide 8).
- Faster (binary compiled in OCaml).
- Retains its advantages on Skolemization-heavy/big proofs.

My Mind Has Changed

Same experiments, but with extracted checker:

Output mode	Skolemization	#problems	Med. time	Avg. time	#faster
Syntactic	Outer	172	0.24s	1.74s	0
TableauxRocq			0.72s	1.91s	0
Extracted			0.003s	0.008s	172
Syntactic	Inner	177	0.24s	1.7s	0
TableauxRocq			0.71s	1.38s	0
Extracted			0.003s	0.006s	177

My Mind Has Changed

Same experiments, but with extracted checker:

Output mode	Skolemization	#problems	Med. time	Avg. time	#faster
Syntactic	Outer	172	0.24s	1.74s	0
TableauxRocq			0.72s	1.91s	0
Extracted			0.003s	0.008s	172
Syntactic	Inner	177	0.24s	1.7s	0
TableauxRocq			0.71s	1.38s	0
Extracted			0.003s	0.006s	177

Extracted checker is sound and fast!

$\rightarrow$ seems like the best way (to date) to certify ATPs outputs.

Conclusion and Future Work

In this work:

- first formalization of free-variable tableaux and their semantic soundness,
- provide sound proof-checker,
- proof output in Goéland, yielding promising benchs,
- difficulty: find the right definition and provide a good API.

Conclusion and Future Work

In this work:

- first formalization of free-variable tableaux and their semantic soundness,
- provide sound proof-checker,
- proof output in Goéland, yielding promising benchs,
- difficulty: find the right definition and provide a good API.

Near future:

- improve internal API with RocqInterface,[3]
- inductive specification of the algorithm,
- implement equality reasoning,
- prove completeness.

[3] c.f., the Rocqshop talk of Saturday.

Conclusion and Future Work

In this work:

- first formalization of free-variable tableaux and their semantic soundness,
- provide sound proof-checker,
- proof output in Goéland, yielding promising benchs,
- difficulty: find the right definition and provide a good API.

Near future:

- improve internal API with RocqInterface,[3]
- inductive specification of the algorithm,
- implement equality reasoning,
- prove completeness.

Available at https://github.com/jrosain/TableauxRocq

[3] c.f., the Rocqshop talk of Saturday.