A formally verified construction of propositional quantifiers for intuitionistic logic

Sam van Gool

IRIF, Université Paris Cité
joint work with Hugo Férée

LLAMA seminar
University of Amsterdam
28 October 2022
Overview

- Aim: verified implementation of Pitts’ interpretation of propositional quantifiers in intuitionistic logic.
Overview

- Aim: verified implementation of Pitts’ interpretation of propositional quantifiers in intuitionistic logic.
- Results:
Overview

- **Aim:** verified implementation of Pitts’ interpretation of propositional quantifiers in intuitionistic logic.
- **Results:**
  - OCaml program calculating propositional quantifiers
Overview

- **Aim:** verified implementation of Pitts’ interpretation of propositional quantifiers in intuitionistic logic.
- **Results:**
  - OCaml program calculating propositional quantifiers
  - Coq proof of correctness
Overview

- **Aim:** verified implementation of Pitts’ interpretation of propositional quantifiers in intuitionistic logic.
- **Results:**
  - OCaml program calculating propositional quantifiers
  - Coq proof of correctness
  - Experimental calculations of propositional quantified formulas
Overview

- **Aim:** verified implementation of Pitts’ interpretation of propositional quantifiers in intuitionistic logic.
- **Results:**
  - OCaml program calculating propositional quantifiers
  - Coq proof of correctness
  - Experimental calculations of propositional quantified formulas
  - Some new theoretical insights on the proof
Propositional quantification

Let $\varphi(p, \bar{q})$ be a propositional formula.
Propositional quantification

- Let $\varphi(p, \bar{q})$ be a propositional formula.
- What are the logical consequences of $\varphi$ that do not contain $p$?
Propositional quantification

Let $\varphi(p, \bar{q})$ be a propositional formula.

What are the logical consequences of $\varphi$ that do not contain $p$?

In classical logic, we always have

$$\varphi \vdash E_p \varphi,$$

where

$$E_p \varphi := \varphi[T/p] \lor \varphi[\bot/p].$$
Propositional quantification

Let $\varphi(p, \bar{q})$ be a propositional formula.

What are the logical consequences of $\varphi$ that do not contain $p$?

In classical logic, we always have

$$\varphi \vdash_{\mathcal{C}} E_p \varphi,$$

where

$$E_p \varphi := \varphi[\top/p] \lor \varphi[\bot/p].$$

Moreover, this is “all there is”, in the following sense:

For any $\psi(\bar{q})$ such that $\varphi \vdash_{\mathcal{C}} \psi$, we also have $E_p \varphi \vdash_{\mathcal{C}} \psi.$
Propositional quantification

- Let \( \varphi(p, \bar{q}) \) be a propositional formula.
- What are the logical consequences of \( \varphi \) that do not contain \( p \)?
- In classical logic, we always have

\[
\varphi \vdash_{c} E_{p}\varphi,
\]

where

\[
E_{p}\varphi := \varphi[\top/p] \lor \varphi[\bot/p].
\]

- Moreover, this is “all there is”, in the following sense:

For any \( \psi(\bar{q}) \) such that \( \varphi \vdash_{c} \psi \), we also have \( E_{p}\varphi \vdash_{c} \psi \).

- The set \( \{ \varphi[\top/p], \varphi[\bot/p] \} \) is a finite basis for the set of \( p \)-free consequences of \( \varphi \).
Quantifiers as adjoints

- For a set of variables $\overline{q}$, denote by $F_{BA}(\overline{q})$ the free Boolean algebra over $\overline{q}$. 

- Elements of $F_{BA}(\overline{q})$ may be represented as formulas $\phi(\overline{q})$ up to $\vdash C$-equivalence.

- The above definition of $E_p \phi$ gives a lower adjoint to the inclusion homomorphism $i: F_{BA}(p, \overline{q}) \rightarrow F_{BA}(\overline{q})$.

- That is, the function $E_p: F_{BA}(p, \overline{q}) \rightarrow F_{BA}(\overline{q})$ is such that, for every $\phi \in F_{BA}(p, \overline{q})$ and $\psi \in F_{BA}(\overline{q})$, $\phi \leq i(\psi) \iff E_p(\phi) \leq \psi$.

- The function $i$ also has an upper adjoint $A_p$, defined by $A_p \phi := \phi[\top/p] \land \phi[\bot/p]$. 


Quantifiers as adjoints

- For a set of variables $\overline{q}$, denote by $F_{BA}(\overline{q})$ the free Boolean algebra over $\overline{q}$.
- Elements of $F_{BA}(\overline{q})$ may be represented as formulas $\varphi(\overline{q})$ up to $\vdash_C$-equivalence.
Quantifiers as adjoints

- For a set of variables $\overline{q}$, denote by $F_{BA}(\overline{q})$ the free Boolean algebra over $\overline{q}$.
- Elements of $F_{BA}(\overline{q})$ may be represented as formulas $\varphi(\overline{q})$ up to $\vdash_C$-equivalence.
- The above definition of $E_p \varphi$ gives a lower adjoint to the inclusion homomorphism $i: F_{BA}(\overline{q}) \to F_{BA}(p, \overline{q})$. 
Quantifiers as adjoints

- For a set of variables $\overline{q}$, denote by $F_{BA}(\overline{q})$ the free Boolean algebra over $\overline{q}$.
- Elements of $F_{BA}(\overline{q})$ may be represented as formulas $\varphi(\overline{q})$ up to $\vdash_C$-equivalence.
- The above definition of $E_p \varphi$ gives a lower adjoint to the inclusion homomorphism $i: F_{BA}(\overline{q}) \to F_{BA}(p, \overline{q})$.
- That is, the function $E_p: F_{BA}(p, \overline{q}) \to F_{BA}(\overline{q})$ is such that, for every $\varphi \in F_{BA}(p, \overline{q})$ and $\psi \in F_{BA}(\overline{q})$,

$$\varphi \leq i(\psi) \iff E_p(\varphi) \leq \psi.$$
Quantifiers as adjoints

- For a set of variables $\overline{q}$, denote by $F_{BA}(\overline{q})$ the free Boolean algebra over $\overline{q}$.
- Elements of $F_{BA}(\overline{q})$ may be represented as formulas $\varphi(\overline{q})$ up to $\vdash_C$-equivalence.
- The above definition of $E_p\varphi$ gives a lower adjoint to the inclusion homomorphism $i: F_{BA}(\overline{q}) \to F_{BA}(p, \overline{q})$.
- That is, the function $E_p: F_{BA}(p, \overline{q}) \to F_{BA}(\overline{q})$ is such that, for every $\varphi \in F_{BA}(p, \overline{q})$ and $\psi \in F_{BA}(\overline{q})$,

$$\varphi \leq i(\psi) \iff E_p(\varphi) \leq \psi.$$ 

- The function $i$ also has an upper adjoint $A_p$, defined by

$$A_p\varphi := \varphi[\top/p] \land \varphi[\bot/p].$$
Pitts’ theorem

- Surprisingly, the same is true in intuitionistic logic.
Pitts’ theorem

- Surprisingly, the same is true in intuitionistic logic.
- Write $F_{HA}(\overline{q})$ for the free Heyting algebra over $\overline{q}$. 

Theorem (Pitts, 1992)

For any finite set of variables $q \cup \{p\}$, the inclusion $i: F_{HA}(q) \rightarrow F_{HA}(p, q)$ has a lower and an upper adjoint.

Concretely, this means that for every $\phi \in F_{HA}(p, q)$, there exist $E_p\phi$ and $A_p\phi$ in $F_{HA}(q)$ such that:

1. $\phi \vdash I E_p\phi$ and for any $\psi \in F_{HA}(q)$, if $\phi \vdash I \psi$ then $E_p\phi \vdash I \psi$,
2. $A_p\phi \vdash I \phi$ and for any $\theta \in F_{HA}(q)$, if $\theta \vdash I \phi$ then $\theta \vdash I A_p\phi$.

$A_p$ and $E_p$ are interpretations of the second order quantifiers $\forall p$ and $\exists p$ in the propositional fragment.
Pitts’ theorem

- Surprisingly, the same is true in intuitionistic logic.
- Write $F_{HA}(\overline{q})$ for the free Heyting algebra over $\overline{q}$.

**Theorem (Pitts, 1992)**

*For any finite set of variables $\overline{q} \cup \{p\}$, the inclusion*

$$i: F_{HA}(\overline{q}) \rightarrow F_{HA}(p, \overline{q})$$

*has a lower and an upper adjoint.*
Pitts’ theorem

- Surprisingly, the same is true in intuitionistic logic.
- Write $F_{HA}(\overline{q})$ for the free Heyting algebra over $\overline{q}$.

**Theorem (Pitts, 1992)**

*For any finite set of variables $\overline{q} \cup \{p\}$, the inclusion*

$$i: F_{HA}(\overline{q}) \to F_{HA}(p, \overline{q})$$

*has a lower and an upper adjoint.*

- Concretely, this means that for every $\varphi \in F_{HA}(p, \overline{q})$, there exist $E_p \varphi$ and $A_p \varphi$ in $F_{HA}(\overline{q})$ such that:
Pitts’ theorem

- Surprisingly, the same is true in intuitionistic logic.
- Write $F_{HA}(\bar{q})$ for the free Heyting algebra over $\bar{q}$.

Theorem (Pitts, 1992)

For any finite set of variables $\bar{q} \cup \{p\}$, the inclusion

$$i: F_{HA}(\bar{q}) \rightarrow F_{HA}(p, \bar{q})$$

has a lower and an upper adjoint.

- Concretely, this means that for every $\varphi \in F_{HA}(p, \bar{q})$, there exist $E_p\varphi$ and $A_p\varphi$ in $F_{HA}(\bar{q})$ such that:
  
  1. $\varphi \vdash \bot \ E_p\varphi$ and for any $\psi \in F_{HA}(\bar{q})$, if $\varphi \vdash \psi$ then $E_p\varphi \vdash \psi$, 


Pitts’ theorem

- Surprisingly, the same is true in intuitionistic logic.
- Write $F_{HA}(\overline{q})$ for the free Heyting algebra over $\overline{q}$.

**Theorem (Pitts, 1992)**

*For any finite set of variables $\overline{q} \cup \{p\}$, the inclusion*

\[ i: F_{HA}(\overline{q}) \to F_{HA}(p, \overline{q}) \]

*has a lower and an upper adjoint.*

- Concretely, this means that for every $\varphi \in F_{HA}(p, \overline{q})$, there exist $E_p \varphi$ and $A_p \varphi$ in $F_{HA}(\overline{q})$ such that:
  1. $\varphi \vdash \underline{I} E_p \varphi$ and for any $\psi \in F_{HA}(\overline{q})$, if $\varphi \vdash \underline{I} \psi$ then $E_p \varphi \vdash \underline{I} \psi$,
  2. $A_p \varphi \vdash \underline{I} \varphi$ and for any $\theta \in F_{HA}(\overline{q})$, if $\theta \vdash \underline{I} \varphi$ then $\theta \vdash \underline{I} A_p \varphi$. 
Pitts’ theorem

- Surprisingly, the same is true in intuitionistic logic.
- Write $F_{HA}(\overline{q})$ for the free Heyting algebra over $\overline{q}$.

**Theorem (Pitts, 1992)**

*For any finite set of variables $\overline{q} \cup \{p\}$, the inclusion*

$$i: F_{HA}(\overline{q}) \rightarrow F_{HA}(p, \overline{q})$$

*has a lower and an upper adjoint.*

- Concretely, this means that for every $\varphi \in F_{HA}(p, \overline{q})$, there exist $E_p\varphi$ and $A_p\varphi$ in $F_{HA}(\overline{q})$ such that:
  1. $\varphi \vdash \downarrow E_p\varphi$ and for any $\psi \in F_{HA}(\overline{q})$, if $\varphi \vdash \downarrow \psi$ then $E_p\varphi \vdash \downarrow \psi$,
  2. $A_p\varphi \vdash \downarrow \varphi$ and for any $\theta \in F_{HA}(\overline{q})$, if $\theta \vdash \downarrow \varphi$ then $\theta \vdash \downarrow A_p\varphi$.
- $A_p$ and $E_p$ are *interpretations* of the second order quantifiers $\forall p$ and $\exists p$ in the propositional fragment.
“Some ten or so years ago I tried to prove the negation of Theorem 1 in connection with (...) the question of whether any Heyting algebra can appear as the algebra of truth-values of an elementary topos. I established that the free Heyting algebra on a countable infinity of generators does not so appear provided the property of $\text{IpC}$ given in Theorem 1 does not hold. It seemed likely to me (and to others to whom I posed the question) that a [formula] $\varphi$ could be found for which $A_p \varphi$ does not exist (although I could not find one!), thus settling the original question about toposes and Heyting algebras in the negative. That Theorem 1 is true is quite a surprise to me. (...) It remains an open question whether every Heyting algebra can be the Lindenbaum algebra of a theory in intuitionistic higher order logic.”

Pitts (1992), p. 36
Uniform interpolation

Combined with the Craig interpolation theorem for IPC, we get:

**Corollary (Uniform interpolation)**

*For any formula* $\varphi(p, q)$, *there exist formulas* $E_p\varphi$ *and* $A_p\varphi$ *such that, for any formula* $\psi(r, q)$,

if $\varphi \vdash \psi$ then $\varphi \vdash E_p\varphi \vdash \psi$,

and

if $\psi \vdash \varphi$ then $\psi \vdash A_p\varphi \vdash \varphi.$
Proofs of Pitts’ theorem

- “Syntactic” (Pitts)
Proofs of Pitts’ theorem

▷ “Syntactic” (Pitts)
  ▸ inductive definition of $E_p\varphi$ and $A_p\varphi$ based on a custom notion of formula weight;
Proofs of Pitts’ theorem

▶ “Syntactic” (Pitts)
  ▶ inductive definition of $E_p\varphi$ and $A_p\varphi$ based on a custom notion of formula weight;
  ▶ correctness of definition using terminating proof calculus $G4ip$. 

▶ "Semantic" (Ghilardi & Zawadowski, also see vG & Reggio):
  ▶ define $[E_p\varphi]$ and $[A_p\varphi]$ as sets of finite Kripke models, or as closed up-sets in the Esakia dual space of $FHA(q)$;
  ▶ by induction on implication depth of $\varphi$, show that the sets of models are definable, or open subsets of the space.

▶ The semantic proofs only give a rough bound on the implication depth of $E_p\varphi$ and $A_p\varphi$, but no direct construction of the formulas.

▶ We implement the syntactic proof.
Proofs of Pitts’ theorem

- “Syntactic” (Pitts)
  - inductive definition of $E_p\varphi$ and $A_p\varphi$ based on a custom notion of formula weight;
  - correctness of definition using terminating proof calculus $G4ip$.
- “Semantic” (Ghilardi & Zawadowski, also see vG & Reggio):
  - define $[E_p\varphi]$ and $[A_p\varphi]$ as sets of finite Kripke models, or as closed up-sets in the Esakia dual space of $FHA(q)$;
  - by induction on implication depth of $\varphi$, show that the sets of models are definable, or open subsets of the space.
  - The semantic proofs only give a rough bound on the implication depth of $E_p\varphi$ and $A_p\varphi$, but no direct construction.
  - We implement the syntactic proof.
Proofs of Pitts’ theorem

- “Syntactic” (Pitts)
  - inductive definition of $E_p\varphi$ and $A_p\varphi$ based on a custom notion of formula weight;
  - correctness of definition using terminating proof calculus $G4ip$.

- “Semantic” (Ghilardi & Zawadowski, also see vG & Reggio):
  - define $[E_p\varphi]$ and $[A_p\varphi]$ as sets of finite Kripke models, or as closed up-sets in the Esakia dual space of $F_{HA}(\vec{q})$;
Proofs of Pitts’ theorem

▸ “Syntactic” (Pitts)
  ▸ inductive definition of \( E_p \varphi \) and \( A_p \varphi \) based on a custom notion of formula \textit{weight};
  ▸ correctness of definition using terminating proof calculus \textbf{G4ip}.

▸ “Semantic” (Ghilardi & Zawadowski, also see vG & Reggio):
  ▸ define \([E_p \varphi]\) and \([A_p \varphi]\) as sets of finite Kripke models, or as closed up-sets in the Esakia dual space of \( F_{HA}(\overline{q}) \);
  ▸ by induction on implication depth of \( \varphi \), show that the sets of models are definable, or open subsets of the space.
Proofs of Pitts’ theorem

- “Syntactic” (Pitts)
  - inductive definition of $E_p\varphi$ and $A_p\varphi$ based on a custom notion of formula *weight*;
  - correctness of definition using terminating proof calculus $\text{G4ip}$.

- “Semantic” (Ghilardi & Zawadowski, also see vG & Reggio):
  - define $[E_p\varphi]$ and $[A_p\varphi]$ as sets of finite Kripke models, or as closed up-sets in the Esakia dual space of $F_{HA}(\overline{q})$;
  - by induction on implication depth of $\varphi$, show that the sets of models are definable, or open subsets of the space.

- The semantic proofs only give a rough bound on the implication depth of $E_p\varphi$ and $A_p\varphi$, but no direct construction of the formulas.
Proofs of Pitts’ theorem

▶ “Syntactic” (Pitts)
  ▶ inductive definition of $E_p\varphi$ and $A_p\varphi$ based on a custom notion of formula weight;
  ▶ correctness of definition using terminating proof calculus $G4ip$.

▶ “Semantic” (Ghilardi & Zawadowski, also see vG & Reggio):
  ▶ define $[E_p\varphi]$ and $[A_p\varphi]$ as sets of finite Kripke models, or as closed up-sets in the Esakia dual space of $F_{HA}(\bar{q})$;
  ▶ by induction on implication depth of $\varphi$, show that the sets of models are definable, or open subsets of the space.

▶ The semantic proofs only give a rough bound on the implication depth of $E_p\varphi$ and $A_p\varphi$, but no direct construction of the formulas.

▶ We implement the syntactic proof.
The proof calculus G4ip

Termination of proof search in Gentzen calculus \textbf{LJ} is not immediate, because of the left implication rule:

\[
\Gamma, \varphi_1 \rightarrow \varphi_2 \vdash \varphi_1 \quad \Gamma, \varphi_2 \vdash \psi \quad \frac{}{\Gamma, \varphi_1 \rightarrow \varphi_2 \vdash \psi}
\]
The proof calculus G4ip

▶ Termination of proof search in Gentzen calculus LJ is not immediate, because of the left implication rule:

\[
\frac{\Gamma, \varphi_1 \rightarrow \varphi_2 \vdash \varphi_1 \quad \Gamma, \varphi_2 \vdash \psi}{\Gamma, \varphi_1 \rightarrow \varphi_2 \vdash \psi}
\]

▶ Natural idea: replace this rule by a finer case analysis, based on the shape of \( \varphi_1 \). For example the rule \((\land \rightarrow L)\):

\[
\frac{\Gamma, (\varphi_1 \rightarrow (\varphi_2 \rightarrow \varphi_3)) \vdash \psi}{\Gamma, ((\varphi_1 \land \varphi_2) \rightarrow \varphi_3) \vdash \psi}
\]
The proof calculus G4ip

- Termination of proof search in Gentzen calculus LJ is not immediate, because of the left implication rule:
  \[
  \Gamma, \varphi_1 \rightarrow \varphi_2 \vdash \varphi_1 \quad \Gamma, \varphi_2 \vdash \psi \\
  \frac{}{\Gamma, \varphi_1 \rightarrow \varphi_2 \vdash \psi}
  \]

- Natural idea: replace this rule by a finer case analysis, based on the shape of \(\varphi_1\). For example the rule \((\land \rightarrow L)\):
  \[
  \Gamma, (\varphi_1 \rightarrow (\varphi_2 \rightarrow \varphi_3)) \vdash \psi \\
  \frac{}{\Gamma, ((\varphi_1 \land \varphi_2) \rightarrow \varphi_3) \vdash \psi}
  \]

- To avoid contraction, use multisets of formulas on the left. The resulting sequent calculus is called G4ip or LJT.
The proof calculus G4ip

- Termination of proof search in Gentzen calculus \( \text{LJ} \) is not immediate, because of the left implication rule:

\[
\frac{\Gamma, \varphi_1 \rightarrow \varphi_2 \vdash \varphi_1 \quad \Gamma, \varphi_2 \vdash \psi}{\Gamma, \varphi_1 \rightarrow \varphi_2 \vdash \psi}
\]

- Natural idea: replace this rule by a finer case analysis, based on the shape of \( \varphi_1 \). For example the rule \((\wedge \rightarrow L)\):

\[
\frac{\Gamma, \varphi_1 \rightarrow (\varphi_2 \rightarrow \varphi_3) \vdash \psi}{\Gamma, ((\varphi_1 \wedge \varphi_2) \rightarrow \varphi_3) \vdash \psi}
\]

- To avoid contraction, use multisets of formulas on the left.

The resulting sequent calculus is called G4ip or LJT.

Theorem (Vorob’ev, Hudelmaier, Dyckhoff)

The sequent calculus G4ip admits contraction and cut and is sound and complete for intuitionistic logic.
**G4ip-provability as an inductive predicate**

▶ What is a **G4ip**-proof of $\Gamma \vdash \varphi$?
What is a \textbf{G4ip}-proof of $\Gamma \vdash \varphi$?  

A proof is a certain tree $p$ labeled by sequents. We say $p$ is a \textit{proof of $\Gamma \vdash \varphi$}, where this is the label of the root of $p$. 
**G4ip-provability as an inductive predicate**

What is a **G4ip**-proof of $\Gamma \vdash \varphi$?

A proof is a certain tree $p$ labeled by sequents. We say $p$ is a *proof of* $\Gamma \vdash \varphi$, where this is the label of the root of $p$.

The set of *proofs* is inductively defined as the smallest set of labeled trees closed under the following rules:
G4ip-provability as an inductive predicate

What is a G4ip-proof of $\Gamma \vdash \varphi$?

A proof is a certain tree $p$ labeled by sequents. We say $p$ is a proof of $\Gamma \vdash \varphi$, where this is the label of the root of $p$.

The set of proofs is inductively defined as the smallest set of labeled trees closed under the following rules:

- (At) for any variable $p$, a single node $(\Gamma, p \vdash p)$ is a proof;
G4ip-provability as an inductive predicate

What is a G4ip-proof of $\Gamma \vdash \varphi$?

A proof is a certain tree $p$ labeled by sequents. We say $p$ is a proof of $\Gamma \vdash \varphi$, where this is the label of the root of $p$.

The set of proofs is inductively defined as the smallest set of labeled trees closed under the following rules:

- (At) for any variable $p$, a single node $(\Gamma, p \vdash p)$ is a proof;
- (ExF) if $\bot \in \Gamma$, then a single node $(\Gamma, \bot \vdash \varphi)$ is a proof;
What is a $G4ip$-proof of $\Gamma \vdash \varphi$?

A proof is a certain tree $p$ labeled by sequents. We say $p$ is a proof of $\Gamma \vdash \varphi$, where this is the label of the root of $p$.

The set of proofs is inductively defined as the smallest set of labeled trees closed under the following rules:

- (At) for any variable $p$, a single node $(\Gamma, p \vdash p)$ is a proof;
- (ExF) if $\bot \in \Gamma$, then a single node $(\Gamma, \bot \vdash \varphi)$ is a proof;
- ($\land$R) if $p$ is a proof of $\Gamma \vdash \varphi$ and $q$ is a proof of $\Gamma \vdash \psi$, then the tree with root labeled by $\Gamma \vdash \varphi \land \psi$, and two subtrees $p$ and $q$, is a proof;
**G4ip**-provability as an inductive predicate

- What is a **G4ip**-proof of $\Gamma \vdash \varphi$?
- A proof is a certain tree $p$ labeled by sequents. We say $p$ is a *proof of* $\Gamma \vdash \varphi$, where this is the label of the root of $p$.
- The set of *proofs* is inductively defined as the smallest set of labeled trees closed under the following rules:
  - (At) for any variable $p$, a single node $(\Gamma, p \vdash p)$ is a proof;
  - (ExF) if $\bot \in \Gamma$, then a single node $(\Gamma, \bot \vdash \varphi)$ is a proof;
  - $(\land R)$ if $p$ is a proof of $\Gamma \vdash \varphi$ and $q$ is a proof of $\Gamma \vdash \psi$, then the tree with root labeled by $\Gamma \vdash \varphi \land \psi$, and two subtrees $p$ and $q$, is a proof;
  - $(\ldots)$
G4ip-provability as an inductive predicate

▶ What is a G4ip-proof of $\Gamma \vdash \varphi$?
▶ A proof is a certain tree $p$ labeled by sequents. We say $p$ is a proof of $\Gamma \vdash \varphi$, where this is the label of the root of $p$.
▶ The set of proofs is inductively defined as the smallest set of labeled trees closed under the following rules:
  ▶ (At) for any variable $p$, a single node $(\Gamma, p \vdash p)$ is a proof;
  ▶ (ExF) if $\bot \in \Gamma$, then a single node $(\Gamma, \bot \vdash \varphi)$ is a proof;
  ▶ ($\land$R) if $p$ is a proof of $\Gamma \vdash \varphi$ and $q$ is a proof of $\Gamma \vdash \psi$, then the tree with root labeled by $\Gamma \vdash \varphi \land \psi$, and two subtrees $p$ and $q$, is a proof;
  ▶ ($\ldots$)
  ▶ ($\land \rightarrow$L) if $p$ is a proof of $\Gamma, \varphi_1 \rightarrow (\varphi_2 \rightarrow \varphi_3) \vdash \psi$, then the tree with root labeled by $\Gamma, (\varphi_1 \land \varphi_2) \rightarrow \varphi_3 \vdash \psi$ and subtree $p$ is a proof.
G4ip-provability in Coq

Inductive Provable : env -> form -> Type :=
| Atom : \forall \Gamma p, \Gamma \bullet (Var p) |- (Var p)
| ExFalso : \forall \Gamma \varphi, \Gamma \bullet \bot |- \varphi
| AndR : \forall \Gamma \varphi \psi, \Gamma |- \varphi -> \Gamma |- \psi
  -> \Gamma |- (\varphi \land \psi)

(* ... *)
| ImpLAnd : \forall \Gamma \varphi1 \varphi2 \varphi3 \psi, \Gamma \bullet (\varphi1 \rightarrow (\varphi2 \rightarrow \varphi3)) |- \psi
  -> \Gamma \bullet ((\varphi1 \land \varphi2) \rightarrow \varphi3) |- \psi

(* ... *)
where "\Gamma |- \varphi" := (Provable \Gamma \varphi).
Definition of propositional quantifiers

- The formula $E_p \varphi$ should be the strongest possible $p$-free formula that is a consequence of $\varphi$, and
Definition of propositional quantifiers

- The formula $E_p\varphi$ should be the strongest possible $p$-free formula that is a consequence of $\varphi$, and
- the formula $A_p\varphi$ should be the weakest possible $p$-free formula having $\varphi$ as a consequence.
Definition of propositional quantifiers

- The formula $E_p \phi$ should be the strongest possible $p$-free formula that is a consequence of $\phi$, and
- the formula $A_p \phi$ should be the weakest possible $p$-free formula having $\phi$ as a consequence.
- Pitts’ definitions have the shape

$$E_p \phi := \bigwedge E_p(\phi) \text{ and } A_p \phi := \bigvee A_p(\phi),$$

where $E_p(\phi)$ and $A_p(\phi)$ are finite sets of formulas.
Definition of propositional quantifiers

- The formula $E_p\varphi$ should be the strongest possible $p$-free formula that is a consequence of $\varphi$, and

- the formula $A_p\varphi$ should be the weakest possible $p$-free formula having $\varphi$ as a consequence.

- Pitts’ definitions have the shape

$$E_p\varphi := \bigwedge E_p(\varphi) \text{ and } A_p\varphi := \bigvee A_p(\varphi),$$

where $E_p(\varphi)$ and $A_p(\varphi)$ are finite sets of formulas.

- For the induction to work, $E_p$ and $A_p$ in fact take not a single formula but a finite pointed multiset of formulas as argument.
## Pitts' table

<table>
<thead>
<tr>
<th></th>
<th>$\Delta$ matches:</th>
<th>$\mathcal{E}(\Delta)$ contains:</th>
</tr>
</thead>
<tbody>
<tr>
<td>$E_1$</td>
<td>$\Delta' \bullet q$</td>
<td>$E(\Delta') \land q$</td>
</tr>
<tr>
<td>$E_4$</td>
<td>$\Delta' \bullet (q \rightarrow \delta)$</td>
<td>$q \rightarrow E(\Delta' \bullet \delta)$</td>
</tr>
<tr>
<td>$E_5$</td>
<td>$\Delta'' \bullet p \bullet (p \rightarrow \delta)$</td>
<td>$E(\Delta'' \bullet p \bullet \delta)$</td>
</tr>
<tr>
<td>$E_6$</td>
<td>$\Delta' \bullet (\delta_1 \land \delta_2) \rightarrow \delta_3$</td>
<td>$E(\Delta' \bullet (\delta_1 \rightarrow (\delta_2 \rightarrow \delta_3)))$</td>
</tr>
<tr>
<td>$E_8$</td>
<td>$\Delta' \bullet ((\delta_1 \rightarrow \delta_2) \rightarrow \delta_3)$</td>
<td>$(E(\Delta' \bullet (\delta_2 \rightarrow \delta_3)) \rightarrow A(\Delta' \bullet (\delta_2 \rightarrow \delta_3), \delta_1 \rightarrow \delta_2)) \rightarrow E(\Delta' \bullet \delta_3)$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>$\Delta, \phi$ matches:</th>
<th>$\mathcal{A}(\Delta, \phi)$ contains:</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A_3$</td>
<td>$\Delta' \bullet \delta_1 \lor \delta_2, \phi$</td>
<td>$(E(\Delta' \bullet \delta_1) \rightarrow A(\Delta' \bullet \delta_1, \phi)) \land (E(\Delta' \bullet \delta_2) \rightarrow A(\Delta' \bullet \delta_2, \phi))$</td>
</tr>
<tr>
<td>$A_7$</td>
<td>$\Delta' \bullet (\delta_1 \lor \delta_2) \rightarrow \delta_3, \phi$</td>
<td>$A(\Delta' \bullet (\delta_1 \rightarrow \delta_3) \bullet (\delta_2 \rightarrow \delta_3), \phi)$</td>
</tr>
<tr>
<td>$A_8$</td>
<td>$\Delta' \bullet ((\delta_1 \rightarrow \delta_2) \rightarrow \delta_3), \phi$</td>
<td>$(E(\Delta' \bullet (\delta_2 \rightarrow \delta_3)) \rightarrow A(\Delta' \bullet (\delta_2 \rightarrow \delta_3), (\delta_1 \rightarrow \delta_2))) \land A(\Delta' \bullet \delta_3, \phi)$</td>
</tr>
<tr>
<td>$A_{11}$</td>
<td>$\Delta, \phi_1 \land \phi_2$</td>
<td>$A(\Delta, \phi_1) \land A(\Delta, \phi_2)$</td>
</tr>
<tr>
<td>$A_{12}$</td>
<td>$\Delta, \phi_1 \lor \phi_2$</td>
<td>$A(\Delta, \phi_1) \lor A(\Delta, \phi_2)$</td>
</tr>
<tr>
<td>$A_{13}$</td>
<td>$\Delta, \phi_1 \rightarrow \phi_2$</td>
<td>$E(\Delta \bullet \phi_1, \phi_2) \rightarrow A(\Delta \bullet \phi_1, \phi_2)$</td>
</tr>
</tbody>
</table>

**Table 1.** Excerpt of Pitts' definitions of $\mathcal{E}(\Delta)$ and $\mathcal{A}(\Delta, \phi)$, with respect to a fixed variable $p$. 

Well-foundedness of multisets

- When formalizing a recursive definition like this in Coq, one must prove that it terminates.
Well-foundedness of multisets

- When formalizing a recursive definition like this in Coq, one must prove that it terminates.
- Even in the paper proof, termination is not entirely obvious, and is proved via well-foundedness of the multiset ordering.
Well-foundedness of multisets

- When formalizing a recursive definition like this in Coq, one must prove that it terminates.
- Even in the paper proof, termination is not entirely obvious, and is proved via well-foundedness of the multiset ordering.
- When $<$ is a preorder on $X$, the Dershowitz-Manna ordering, $\prec$, on the set of finite multisets of $X$ is the transitive closure of the one-step relation $S \uplus T \prec S \bullet x$, where $T$ is any finite multiset such that $t < x$ for all $t \in T$. 
Well-foundedness of multisets

- When formalizing a recursive definition like this in Coq, one must prove that it terminates.
- Even in the paper proof, termination is not entirely obvious, and is proved via *well-foundedness of the multiset ordering*.
- When $<$ is a preorder on $X$, the *Dershowitz-Manna* ordering, $\prec$, on the set of finite multisets of $X$ is the transitive closure of the one-step relation $S \uplus T \prec S \bullet x$, where $T$ is any finite multiset such that $t < x$ for all $t \in T$.

**Theorem (Dershowitz, Manna)**

*If the order $<$ on $X$ is well-founded, then $\prec$ on the finite multisets of $X$ is well-founded.*
Define a preorder on the set of formulas $F$ by $\varphi < \psi$ iff $w(\varphi) < w(\psi)$, where $w: F \to \mathbb{N}$ is the weight function, defined by induction on formula complexity.
Proving termination

- Define a preorder on the set of formulas $F$ by $\varphi < \psi$ iff $w(\varphi) < w(\psi)$, where $w: F \to \mathbb{N}$ is the weight function, defined by induction on formula complexity.
- By the Dershowitz-Manna theorem, $\prec$ on finite (pointed) multisets is well-founded.
Proving termination

- Define a preorder on the set of formulas $F$ by $\varphi < \psi$ iff $w(\varphi) < w(\psi)$, where $w: F \rightarrow \mathbb{N}$ is the weight function, defined by induction on formula complexity.
- By the Dershowitz-Manna theorem, $\prec$ on finite (pointed) multisets is well-founded.
- Observe that the applications of $E$ and $A$ in the right column of Pitts’ table always take arguments that are $\prec$-lighter.
Proving termination

- Define a preorder on the set of formulas $F$ by $\varphi < \psi$ iff $w(\varphi) < w(\psi)$, where $w : F \rightarrow \mathbb{N}$ is the weight function, defined by induction on formula complexity.

- By the Dershowitz-Manna theorem, $\prec$ on finite (pointed) multisets is well-founded.

- Observe that the applications of $E$ and $A$ in the right column of Pitts’ table always take arguments that are $\prec$-lighter.

- Formalizing the word “observe” requires a non-trivial amount of meta-programming work in Coq, which I will only sketch.
What’s in a proof?

- Proof assistants like Coq, Lean, etc. take the Curry-Howard correspondence (very) seriously.

\[
\text{Inductive Nat : Type} := \\
\text{Zero : Nat} \mid \\
\text{Succ : Nat \rightarrow Nat}
\]

- and statements about \( \text{Nat} \) are also defined as types:

\[
\text{Definition pluscomm : Type} := \\
\forall a b : \text{Nat}, \ a + b = b + a
\]

- A proof is then a term of this type.
What’s in a proof?

- Proof assistants like Coq, Lean, etc. take the Curry-Howard correspondence (very) seriously.
- Mathematical definitions and statements are *types*.
What’s in a proof?

- Proof assistants like Coq, Lean, etc. take the Curry-Howard correspondence (very) seriously.
- Mathematical definitions and statements are *types*.
- For example, the type \( \mathbb{N} \) is defined as

```plaintext
Inductive Nat : Type :=
| Zero : Nat
| Succ : Nat -> Nat
```
What’s in a proof?

► Proof assistants like Coq, Lean, etc. take the Curry-Howard correspondence (very) seriously.
► Mathematical definitions and statements are *types*.
► For example, the type \( \mathbb{N} \) is defined as

\[
\text{Inductive Nat : Type :=} \\
| \text{Zero : Nat} \\
| \text{Succ : Nat \rightarrow Nat}
\]

► and statements *about* \( \mathbb{N} \) are also defined as types:

\[
\text{Definition pluscomm : Type := } \forall a b : \text{Nat}, a + b = b + a.
\]
What’s in a proof?

- Proof assistants like Coq, Lean, etc. take the Curry-Howard correspondence (very) seriously.
- Mathematical definitions and statements are types.
- For example, the type \( \mathbb{N} \) is defined as

```plaintext
Inductive Nat : Type :=
  | Zero : Nat
  | Succ : Nat \to\ Nat
```

- and statements about \( \mathbb{N} \) are also defined as types:

```plaintext
Definition pluscomm : Type := \forall a b : Nat, a + b = b + a.
```

- A proof is then a term of this type.
Programming

- Proofs, or programs, are written as λ-terms:

```
Definition swap : Nat -> Nat -> (Nat * Nat) :=
    fun a b : Nat => (b, a)
```
Proofs, or programs, are written as λ-terms:

\[
\text{Definition } \text{swap} : \text{Nat} \to \text{Nat} \to (\text{Nat} \times \text{Nat}) := \\
\quad \text{fun } a \ b : \text{Nat} \Rightarrow (b, a)
\]

Programs may be defined by recursion:

\[
\text{Fixpoint } \text{plus} (a : \text{Nat}) (b : \text{Nat}) : \text{Nat} := \\
\quad \text{match } b \text{ with} \\
\quad \mid \text{Zero} \Rightarrow a \\
\quad \mid \text{Succ } n \Rightarrow \text{Succ } (\text{plus } a \ n)
\]

end.
Proofs, or programs, are written as $\lambda$-terms:

**Definition** \( \text{swap} : \text{Nat} \rightarrow \text{Nat} \rightarrow (\text{Nat} \times \text{Nat}) := \)

\[
\text{fun } a \text{ b : Nat } \Rightarrow (b, a)
\]

Programs may be defined by recursion:

**Fixpoint** \( \text{plus} (a : \text{Nat}) (b : \text{Nat}) : \text{Nat} := \)

\[
\text{match b with}
| \text{Zero } \Rightarrow a
| \text{Succ n } \Rightarrow \text{Succ (plus a n)}
\text{end.}
\]

While it is possible to also define \textit{proofs} in this way, even simple proofs become too long to fit on a slide:
The proof that $+$ is commutative

```
fun n m : nat =>
  Nat.bi_induction (fun t : nat => t + m = m + t)
  (((fun (x y : nat) (H : x = y) =>
    Morphisms.trans_co_eq_inv_impl_morphism RelationClasses.iff_Transitive
    (x + m = m + x) (y + m = m + y)
    (Morphisms.PER_morphism (RelationClasses.Equivalence_PER Nat.eq_equiv)
     (x + m) (y + m)
     (Nat.add_wd x y H m m
      (Morphisms.reflexive_proper_proxy
       RelationClasses.Equivalence_Reflexive m)))
    (m + x) (m + y)
    (Morphisms.Reflexive_partial_app_morphism Nat.add_wd
     (Morphisms.reflexive_proper_proxy
      RelationClasses.Equivalence_Reflexive m) x y H))
  (y + m = m + y) (y + m = m + y)
  (Morphisms.eq_proper_proxy (y + m = m + y))
  (RelationClasses.reflexivity (y + m = m + y)))
```

(* ... 40 more lines of code ...*)
To formalize proofs, one usually does not write the proof terms directly, but instead uses tactics.
To formalize proofs, one usually does not write the proof terms directly, but instead uses tactics.

A sequence of tactics is a ‘recipe’ for building a proof.
Metaprogramming

- To formalize proofs, one usually does not write the proof terms directly, but instead uses tactics.
- A sequence of tactics is a ‘recipe’ for building a proof.
- Since a proof is a program, a tactic is a program that produces a program.
Metaprogramming

- To formalize proofs, one usually does not write the proof terms directly, but instead uses tactics.
- A sequence of tactics is a ‘recipe’ for building a proof.
- Since a proof is a program, a tactic is a program that produces a program.
- Writing tactics is therefore ‘metaprogramming’.
Tactics in our proof

► Coming back to our definition of propositional quantifiers:

Program Fixpoint EA (pe : env * form) :=
    let Δ := fst pe in
    (∧ (in_map Δ (e_rule EA)),
    ∨ (in_map Δ (a_rule_env EA)) ⊻ a_rule_form EA).

Next Obligation. apply wf_pointed_order. Defined.
Tactics in our proof

Coming back to our definition of propositional quantifiers:

Program Fixpoint EA (pe : env * form) :=
  let Δ := fst pe in
  (∧ (in_map Δ (e_rule EA)),
   ∨ (in_map Δ (a_rule_env EA)) ⊻ a_rule_form EA).
Next Obligation. apply wf_pointed_order. Defined.

The ‘obligation’ to show that this fixpoint definition
terminates is fulfilled by tactics.
A remark on multisets

- In order not have to also implement the Dershowitz-Manna theorem, we imported it from an existing library ("CoLoR").
A remark on multisets

- In order not have to also implement the Dershowitz-Manna theorem, we imported it from an existing library (“CoLoR”).
- We also imported specific tactics about multisets from another existing library (“IRIS std++”).
A remark on multisets

- In order not have to also implement the Dershowitz-Manna theorem, we imported it from an existing library ("CoLoR").
- We also imported specific tactics about multisets from another existing library ("IRIS std++").
- Some engineering was needed to convince Coq that the notion of ‘multiset’ from two different libraries was the same.
A remark on multisets

- In order not to have to also implement the Dershowitz-Manna theorem, we imported it from an existing library ("CoLoR").
- We also imported specific tactics about multisets from another existing library ("IRIS std++").
- Some engineering was needed to convince Coq that the notion of ‘multiset’ from two different libraries was the same.
- Only after this work was already done, we realized that it may have been simpler to directly define a “weight” on multisets, since Pitts’ table only uses the multiset ordering in a weak way: there is a uniform bound on the step size.
The correctness proof

- The easy parts:

- The hard parts:

- This is by induction on the G4ip-proof of a sequent $\phi \vdash \psi$, distinguishing cases according to the last rule.

- Pitts' proof uses 'obvious' facts about intuitionistic logic, which we however had to verify formally in G4ip.

- Fortunately, the pen-and-paper proofs of admissibility of weakening, contraction, and special cuts had been mostly done by Dyckhoff and Negri, and became $\sim 800$ lines of Coq.

- The proof of the hard part: $\sim 200$ lines of Coq tactics; but the generated proof term is $\sim 5000$ lines.

- Entire development: $\sim 2500$ lines of Coq total.
The correctness proof

- The easy parts:
  - \( p \) does not occur in \( E_p \varphi \) and \( A_p \varphi \);
The correctness proof

- The easy parts:
  - \( p \) does not occur in \( E_p \varphi \) and \( A_p \varphi \);
  - the sequents \( \varphi \vdash E_p \varphi \) and \( A_p \varphi \vdash \varphi \) are provable.

- The hard parts:
  - \( E_p \varphi \) is minimal among \( \varphi \vdash \)
  - \( A_p \varphi \) is maximal among \( \vdash \varphi \).

  This is by induction on the \( G4ip \)-proof of a sequent \( \varphi \vdash \psi \), distinguishing cases according to the last rule.

  Pitts' proof uses 'obvious' facts about intuitionistic logic, which we however had to verify formally in \( G4ip \).

  Fortunately, the pen-and-paper proofs of admissibility of weakening, contraction, and special cuts had been mostly done by Dyckhoff and Negri, and became \( \sim 800 \) lines of Coq.

  The proof of the hard part: \( \sim 200 \) lines of Coq tactics; but the generated proof term is \( \sim 5000 \) lines.

  Entire development: \( \sim 2500 \) lines of Coq total.
The correctness proof

▶ The easy parts:
  ▶ $p$ does not occur in $E_p \varphi$ and $A_p \varphi$;
  ▶ the sequents $\varphi \vdash E_p \varphi$ and $A_p \varphi \vdash \varphi$ are provable.

▶ The hard parts:

$E_p \varphi$ is minimal among $\varphi \vdash$ and $A_p \varphi$ is maximal among $\vdash \varphi$. 
The correctness proof

The easy parts:

- $p$ does not occur in $E_p \varphi$ and $A_p \varphi$;
- the sequents $\varphi \vdash E_p \varphi$ and $A_p \varphi \vdash \varphi$ are provable.

The hard parts:

- $E_p \varphi$ is minimal among $\varphi \vdash$ and $A_p \varphi$ is maximal among $\vdash \varphi$.
- This is by induction on the G4ip-proof of a sequent $\varphi \vdash \psi$, distinguishing cases according to the last rule.

Pitts' proof uses 'obvious' facts about intuitionistic logic, which we however had to verify formally in G4ip.

Fortunately, the pen-and-paper proofs of admissibility of weakening, contraction, and special cuts had been mostly done by Dyckhoff and Negri, and became ∼800 lines of Coq.

The proof of the hard part: ∼200 lines of Coq tactics; but the generated proof term is ∼5000 lines.

Entire development: ∼2500 lines of Coq total.
The correctness proof

The easy parts:
- $p$ does not occur in $E_p\varphi$ and $A_p\varphi$;
- the sequents $\varphi \vdash E_p\varphi$ and $A_p\varphi \vdash \varphi$ are provable.

The hard parts:
- $E_p\varphi$ is minimal among $\varphi \vdash$ and $A_p\varphi$ is maximal among $\vdash \varphi$.
- This is by induction on the G4ip-proof of a sequent $\varphi \vdash \psi$, distinguishing cases according to the last rule.
- Pitts’ proof uses ‘obvious’ facts about intuitionistic logic, which we however had to verify formally in G4ip.
The correctness proof

- The easy parts:
  - $p$ does not occur in $E_p\varphi$ and $A_p\varphi$;
  - the sequents $\varphi \vdash E_p\varphi$ and $A_p\varphi \vdash \varphi$ are provable.

- The hard parts:
  - $E_p\varphi$ is minimal among $\varphi \vdash$ and $A_p\varphi$ is maximal among $\vdash \varphi$.
  - This is by induction on the $\text{G4ip}$-proof of a sequent $\varphi \vdash \psi$, distinguishing cases according to the last rule.
  - Pitts’ proof uses ‘obvious’ facts about intuitionistic logic, which we however had to verify formally in $\text{G4ip}$.
  - Fortunately, the pen-and-paper proofs of admissibility of weakening, contraction, and special cuts had been mostly done by Dyckhoff and Negri, and became $\sim 800$ lines of Coq.
The correctness proof

- The easy parts:
  - $p$ does not occur in $E_p\varphi$ and $A_p\varphi$;
  - the sequents $\varphi \vdash E_p\varphi$ and $A_p\varphi \vdash \varphi$ are provable.

- The hard parts:
  - $E_p\varphi$ is minimal among $\varphi \vdash$ and $A_p\varphi$ is maximal among $\vdash \varphi$.
  - This is by induction on the G4ip-proof of a sequent $\varphi \vdash \psi$, distinguishing cases according to the last rule.
  - Pitts’ proof uses ‘obvious’ facts about intuitionistic logic, which we however had to verify formally in G4ip.
  - Fortunately, the pen-and-paper proofs of admissibility of weakening, contraction, and special cuts had been mostly done by Dyckhoff and Negri, and became $\sim$800 lines of Coq.
  - The proof of the hard part: $\sim$200 lines of Coq tactics; but the generated proof term is $\sim$5000 lines.
The correctness proof

- The easy parts:
  - $p$ does not occur in $E_p \varphi$ and $A_p \varphi$;
  - the sequents $\varphi \vdash E_p \varphi$ and $A_p \varphi \vdash \varphi$ are provable.

- The hard parts:
  - $E_p \varphi$ is minimal among $\varphi \vdash$ and $A_p \varphi$ is maximal among $\vdash \varphi$.
  - This is by induction on the G4ip-proof of a sequent $\varphi \vdash \psi$, distinguishing cases according to the last rule.
  - Pitts' proof uses 'obvious' facts about intuitionistic logic, which we however had to verify formally in G4ip.
  - Fortunately, the pen-and-paper proofs of admissibility of weakening, contraction, and special cuts had been mostly done by Dyckhoff and Negri, and became $\sim 800$ lines of Coq.
  - The proof of the hard part: $\sim 200$ lines of Coq tactics; but the generated proof term is $\sim 5000$ lines.
  - Entire development: $\sim 2500$ lines of Coq total.
Final result

Theorem pitts p V : (p \notin V) ->

\forall \varphi, \text{vars_incl} \varphi (p :: V) ->

(\text{vars_incl} (E \ p \ \varphi) V)

* (\{[\varphi]\} \vdash (E \ p \ \varphi))

* (\forall \psi, \text{vars_incl} \psi V -> \{[\varphi]\} \vdash \psi -> \{[E \ p \ \varphi]\} \vdash \psi)

* (\text{vars_incl} (A \ p \ \varphi) V)

* (\{[A \ p \ \varphi]\} \vdash \varphi)

* (\forall \theta, \text{vars_incl} \theta V -> \{[\theta]\} \vdash \varphi -> \{[\theta]\} \vdash A \ p \ \varphi).
Extraction

- We can in particular extract a program that computes propositional quantifiers of any formula in IPC, guaranteed to be correct.
Extraction

- We can in particular extract a program that computes propositional quantifiers of any formula in IPC, guaranteed to be correct.
- The code is automatically produced from the Coq implementation.
Extraction

- We can in particular extract a program that computes propositional quantifiers of any formula in IPC, guaranteed to be correct.
- The code is automatically produced from the Coq implementation.
- (Demo)
Some first experimental results

Define

\[ \varphi_0 := p_0 \]

\[ \varphi_{n+1} := \varphi_n \rightarrow p_{n+1} \]

<table>
<thead>
<tr>
<th>( n )</th>
<th>weight of ( E_{p_0} \varphi_n )</th>
<th>weight of ( A_{p_0} \varphi_n )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>28</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>28</td>
<td>62</td>
</tr>
<tr>
<td>4</td>
<td>1447</td>
<td>62</td>
</tr>
<tr>
<td>5</td>
<td>1447</td>
<td>2900</td>
</tr>
<tr>
<td>6</td>
<td>152137</td>
<td>2900</td>
</tr>
<tr>
<td>7</td>
<td>(timeout)</td>
<td>(timeout)</td>
</tr>
</tbody>
</table>
Improvements and future work

▶ We did not make much effort to *simplify* the computed formulas, only some of the most basic equivalences, like \( \bot \rightarrow \varphi \equiv \top \), \( \varphi \rightarrow \varphi \equiv \top \), etc.

▶ Potential for more experimentation, a better understanding of the quantifiers.

▶ Generalizations to other logics? (cf. Iemhoff et al.)

▶ A computational interpretation of Pitts' theorem?

▶ Formalizing semantic approaches? (cf. Gattinger, work in progress with Gallego Arias, et al.)

▶ Current version available at: https://samvangool.net/ipq/
Improvements and future work

▶ We did not make much effort to *simplify* the computed formulas, only some of the most basic equivalences, like \( \bot \rightarrow \varphi \equiv \top \), \( \varphi \rightarrow \varphi \equiv \top \), etc.

▶ Potential for more experimentation, a better understanding of the quantifiers.
Improvements and future work

- We did not make much effort to simplify the computed formulas, only some of the most basic equivalences, like $\bot \rightarrow \varphi \equiv \top, \varphi \rightarrow \varphi \equiv \top$, etc.
- Potential for more experimentation, a better understanding of the quantifiers.
- Generalizations to other logics? (cf. Iemhoff et al.)
Improvements and future work

▶ We did not make much effort to simplify the computed formulas, only some of the most basic equivalences, like $\bot \rightarrow \varphi \equiv T$, $\varphi \rightarrow \varphi \equiv T$, etc.

▶ Potential for more experimentation, a better understanding of the quantifiers.

▶ Generalizations to other logics? (cf. Iemhoff et al.)

▶ A computational interpretation of Pitts’ theorem?
Improvements and future work

- We did not make much effort to simplify the computed formulas, only some of the most basic equivalences, like $\bot \rightarrow \varphi \equiv \top, \varphi \rightarrow \varphi \equiv \top$, etc.

- Potential for more experimentation, a better understanding of the quantifiers.

- Generalizations to other logics? (cf. Iemhoff et al.)

- A computational interpretation of Pitts’ theorem?

- Formalizing semantic approaches? (cf. Gattinger, work in progress with Gallego Arias, et al.)
Improvements and future work

- We did not make much effort to simplify the computed formulas, only some of the most basic equivalences, like \( \bot \rightarrow \varphi \equiv \top \), \( \varphi \rightarrow \varphi \equiv \top \), etc.

- Potential for more experimentation, a better understanding of the quantifiers.

- Generalizations to other logics? (cf. Ilemhoff et al.)

- A computational interpretation of Pitts’ theorem?

- Formalizing semantic approaches? (cf. Gattinger, work in progress with Gallego Arias, et al.)

- Current version available at:
  https://samvangool.net/ipq/