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

Propositional quantification

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

Propositional quantification

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

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

where

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

Propositional quantification

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

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

where

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

- Moreover, this is "all there is", in the following sense:
For any $\psi(\overline{q})$ such that $\varphi \vdash_{\mathbf{C}} \psi$, we also have $E_p\varphi \vdash_{\mathbf{C}} \psi$.

Propositional quantification

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

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

where

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

- Moreover, this is "all there is", in the following sense:
For any $\psi(\overline{q})$ such that $\varphi \vdash_{\mathbf{C}} \psi$, we also have $E_p\varphi \vdash_{\mathbf{C}} \psi$.
- The set $\{\varphi[\top/p], \varphi[\bot/p]\}$ is a finite basis for the set of p-free consequences of φ.

Quantifiers as adjoints

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

Quantifiers as adjoints

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

Quantifiers as adjoints

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

Quantifiers as adjoints

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

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

Quantifiers as adjoints

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

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

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

$$A_p\varphi := \varphi[\top/p] \wedge \varphi[\bot/p].$$

Pitts' theorem

- Surprisingly, the same is true in intuitionistic logic.

Pitts' theorem

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

Pitts' theorem

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

Theorem (Pitts, 1992)

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

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

has a lower and an upper adjoint.

Pitts' theorem

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

Theorem (Pitts, 1992)

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

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

has a lower and an upper adjoint.

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

Pitts' theorem

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

Theorem (Pitts, 1992)

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

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

has a lower and an upper adjoint.

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

Pitts' theorem

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

Theorem (Pitts, 1992)

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

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

has a lower and an upper adjoint.

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

Pitts' theorem

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

Theorem (Pitts, 1992)

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

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

has a lower and an upper adjoint.

- Concretely, this means that for every $\varphi \in F_{\mathrm{HA}}(p, \overline{q})$, there exist $E_p\varphi$ and $A_p\varphi$ in $F_{\mathrm{HA}}(\overline{q})$ such that:
    1. $\varphi \vdash_{\mathbf{I}} E_p\varphi$ and for any $\psi \in F_{\mathrm{HA}}(\overline{q})$, if $\varphi \vdash_{\mathbf{I}} \psi$ then $E_p\varphi \vdash_{\mathbf{I}} \psi$,
    2. $A_p\varphi \vdash_{\mathbf{I}} \varphi$ and for any $\theta \in F_{\mathrm{HA}}(\overline{q})$, if $\theta \vdash_{\mathbf{I}} \varphi$ then $\theta \vdash_{\mathbf{I}} A_p\varphi$.
- A_p and E_p are *interpretations* of the second order quantifiers $\forall p$ and $\exists p$ in the propositional fragment.

Origins

"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 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] φ 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 $\varphi(\overline{p}, \overline{q})$, there exist formulas $E_{\bar{p}}\varphi$ and $A_{\bar{p}}\varphi$ such that, for any formula $\psi(\bar{r}, \overline{q})$,

$$\text{if } \varphi \vdash \psi \text{ then } \varphi \vdash E_{\bar{p}}\varphi \vdash \psi,$$

and

$$\text{if } \psi \vdash \varphi \text{ then } \psi \vdash A_{\bar{p}}\varphi \vdash \varphi.$$

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**.

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):

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 $F_{\mathrm{HA}}(\overline{q})$;

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 $F_{\mathrm{HA}}(\overline{q})$;
  - by induction on implication depth of φ, 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 **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 $F_{\mathrm{HA}}(\overline{q})$;
  - by induction on implication depth of φ, 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 $F_{\mathrm{HA}}(\overline{q})$;
  - by induction on implication depth of φ, 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 **LJ** is not immediate, because of the left implication rule:

$$\frac{\Gamma, \varphi_1 \to \varphi_2 \vdash \varphi_1 \qquad \Gamma, \varphi_2 \vdash \psi}{\Gamma, \varphi_1 \to \varphi_2 \vdash \psi}$$

The proof calculus G4ip

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

$$\frac{\Gamma, \varphi_1 \to \varphi_2 \vdash \varphi_1 \qquad \Gamma, \varphi_2 \vdash \psi}{\Gamma, \varphi_1 \to \varphi_2 \vdash \psi}$$

- Natural idea: replace this rule by a finer case analysis, based on the shape of φ_1. For example the rule ($\wedge \to$L):

$$\frac{\Gamma, (\varphi_1 \to (\varphi_2 \to \varphi_3)) \vdash \psi}{\Gamma, ((\varphi_1 \wedge \varphi_2) \to \varphi_3) \vdash \psi}$$

The proof calculus G4ip

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

$$\frac{\Gamma, \varphi_1 \to \varphi_2 \vdash \varphi_1 \qquad \Gamma, \varphi_2 \vdash \psi}{\Gamma, \varphi_1 \to \varphi_2 \vdash \psi}$$

- Natural idea: replace this rule by a finer case analysis, based on the shape of φ_1. For example the rule ($\wedge \to$L):

$$\frac{\Gamma, (\varphi_1 \to (\varphi_2 \to \varphi_3)) \vdash \psi}{\Gamma, ((\varphi_1 \wedge \varphi_2) \to \varphi_3) \vdash \psi}$$

- 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 **LJ** is not immediate, because of the left implication rule:

$$\frac{\Gamma, \varphi_1 \to \varphi_2 \vdash \varphi_1 \qquad \Gamma, \varphi_2 \vdash \psi}{\Gamma, \varphi_1 \to \varphi_2 \vdash \psi}$$

- Natural idea: replace this rule by a finer case analysis, based on the shape of φ_1. For example the rule ($\wedge \to$L):

$$\frac{\Gamma, (\varphi_1 \to (\varphi_2 \to \varphi_3)) \vdash \psi}{\Gamma, ((\varphi_1 \wedge \varphi_2) \to \varphi_3) \vdash \psi}$$

- 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 $\Gamma \vdash \varphi$?

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.

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;

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;
  - ($\wedge$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 $\Gamma \vdash \varphi \wedge \psi$, 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;
  - ($\wedge$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 $\Gamma \vdash \varphi \wedge \psi$, 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;
  - ($\wedge$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 $\Gamma \vdash \varphi \wedge \psi$, and two subtrees p and q, is a proof;
  - $(\ldots)$
  - ($\wedge \to$L) if p is a proof of $\Gamma, \varphi_1 \to (\varphi_2 \to \varphi_3) \vdash \psi$, then the tree with root labeled by $\Gamma, (\varphi_1 \wedge \varphi_2) \to \varphi_3 \vdash \psi$ and subtree p is a proof.

G4ip-provability in Coq

```
Inductive Provable : env -> form -> Type :=
| Atom :     ∀ Γ p, Γ • (Var p) ⊢ (Var p)
| ExFalso : ∀ Γ φ, Γ • ⊥ ⊢ φ
| AndR :     ∀ Γ φ ψ, Γ ⊢ φ -> Γ ⊢ ψ
                     -> Γ ⊢ (φ ∧ ψ)
(* ... *)
| ImpLAnd : ∀ Γ φ1 φ2 φ3 ψ, Γ • (φ1 → (φ2 → φ3)) ⊢ ψ
                           -> Γ • ((φ1 ∧ φ2) → φ3)  ⊢ ψ
(* ... *)
where "Γ ⊢ φ" := (Provable Γ φ).
```

Definition of propositional quantifiers

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

Definition of propositional quantifiers

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

Definition of propositional quantifiers

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

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

where $\mathcal{E}_p(\varphi)$ and $\mathcal{A}_p(\varphi)$ 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 φ, and
- the formula $A_p\varphi$ should be the weakest possible p-free formula having φ as a consequence.
- Pitts' definitions have the shape

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

  where $\mathcal{E}_p(\varphi)$ and $\mathcal{A}_p(\varphi)$ are *finite* sets of formulas.
- For the induction to work, $\mathcal{E}_p$ and $\mathcal{A}_p$ in fact take not a single formula but a *finite pointed multiset* of formulas as argument.

Pitts' table

	Δ matches:	$\mathcal{E}(\Delta)$ contains:
E_1	$\Delta' \bullet q$	$E(\Delta') \wedge q$
E_4	$\Delta' \bullet (q \to \delta)$	$q \to E(\Delta' \bullet \delta)$
E_5	$\Delta'' \bullet p \bullet (p \to \delta)$	$E(\Delta'' \bullet p \bullet \delta)$
E_6	$\Delta' \bullet (\delta_1 \wedge \delta_2) \to \delta_3$	$E(\Delta' \bullet (\delta_1 \to (\delta_2 \to \delta_3)))$
E_8	$\Delta' \bullet ((\delta_1 \to \delta_2) \to \delta_3)$	$(E(\Delta' \bullet (\delta_2 \to \delta_3)) \to A(\Delta' \bullet (\delta_2 \to \delta_3), \delta_1 \to \delta_2)) \to E(\Delta' \bullet \delta_3)$

	Δ, ϕ matches:	$\mathcal{A}(\Delta, \phi)$ contains:
A_3	$\Delta' \bullet \delta_1 \vee \delta_2, \phi$	$(E(\Delta' \bullet \delta_1) \to A(\Delta' \bullet \delta_1, \phi)) \wedge (E(\Delta' \bullet \delta_2) \to A(\Delta' \bullet \delta_2, \phi))$
A_7	$\Delta' \bullet (\delta_1 \vee \delta_2) \to \delta_3, \phi$	$A(\Delta' \bullet (\delta_1 \to \delta_3) \bullet (\delta_2 \to \delta_3), \phi)$
A_8	$\Delta' \bullet ((\delta_1 \to \delta_2) \to \delta_3), \phi$	$(E(\Delta' \bullet (\delta_2 \to \delta_3)) \to A(\Delta' \bullet (\delta_2 \to \delta_3), (\delta_1 \to \delta_2))) \wedge A(\Delta' \bullet \delta_3, \phi)$
A_{11}	$\Delta, \phi_1 \wedge \phi_2$	$A(\Delta, \phi_1) \wedge A(\Delta, \phi_2)$
A_{12}	$\Delta, \phi_1 \vee \phi_2$	$A(\Delta, \phi_1) \vee A(\Delta, \phi_2)$
A_{13}	$\Delta, \phi_1 \to \phi_2$	$E(\Delta \bullet \phi_1, \phi_2) \to A(\Delta \bullet \phi_1, \phi_2)$

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.

Proving termination

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

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

```
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

```
Inductive Nat : Type :=
| Zero : Nat
| Succ : Nat -> Nat
```

- and statements *about* $\mathbb{N}$ are *also* defined as types:

```
Definition pluscomm : Type := ∀ a b : 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

```
Inductive Nat : Type :=
| Zero : Nat
| Succ : Nat -> Nat
```

- and statements *about* $\mathbb{N}$ are *also* defined as types:

```
Definition pluscomm : Type := ∀ 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)
```

Programming

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

```
Definition swap : Nat -> Nat -> (Nat * Nat) :=
                  fun a b : Nat => (b, a)
```

- Programs may be defined by recursion:

```
Fixpoint plus (a : Nat) (b : Nat) : Nat :=
    match b with
    | Zero => a
    | Succ n => Succ (plus a n)
    end.
```

Programming

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

```
Definition swap : Nat -> Nat -> (Nat * Nat) :=
                  fun a b : Nat => (b, a)
```

- Programs may be defined by recursion:

```
Fixpoint plus (a : Nat) (b : Nat) : Nat :=
    match b with
    | Zero => a
    | Succ n => Succ (plus a n)
    end.
```

- While it is possible to also define *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 ...*)
```

Metaprogramming

- To formalize proofs, one usually does not write the proof terms directly, but instead uses *tactics*.

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.

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 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 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 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.

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* **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 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 ∉ V) ->
  ∀ φ, vars_incl φ (p :: V) ->
    (vars_incl (E p φ) V)
  * ({[φ]} ⊢ (E p φ))
  * (∀ ψ, vars_incl ψ V -> {[φ]} ⊢ ψ -> {[E p φ]} ⊢ ψ)
  * (vars_incl (A p φ) V)
  * ({[A p φ]} ⊢ φ)
  * (∀ θ, vars_incl θ V -> {[θ]} ⊢ φ -> {[θ]} ⊢ A p φ).
```

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

$$\varphi_0 := p_0$$

$$\varphi_{n+1} := \varphi_n \to p_{n+1}$$

n	weight of $E_{p_0}\varphi_n$	weight of $A_{p_0}\varphi_n$
1	3	5
2	28	5
3	28	62
4	1447	62
5	1447	2900
6	152137	2900
7	(timeout)	(timeout)

Improvements and future work

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

Improvements and future work

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

Improvements and future work

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