Gödel's functional interpretation and the concept of learning

Thomas Powell

University of Innsbruck

Logic in Computer Science (LICS'16)

Columbia University, New York
United States of America
5 July 2016

What is the computational meaning of classical reasoning?

What is the computational meaning of classical reasoning?

One could consider the 'finitized' version of classical theorems:

A : there exists an ideal object x.

A_{fin} : there exist finite approximations to x of arbitrary high quality.

What is the computational meaning of classical reasoning?

One could consider the 'finitized' version of classical theorems:

A : there exists an ideal object x.

A_{fin} : there exist finite approximations to x of arbitrary high quality.

Over classical logic, $A \leftrightarrow A_{\text{fin}}$.

While ideal objects cannot be effectively constructed, finite approximations to them can.

This talk is about *algorithms* which compute such approximations.

What is the computational meaning of a Π_3-theorem?

$$A :\equiv \forall x^X \exists y^Y \forall z^Z P(x, y, z)$$

What is the computational meaning of a Π_3-theorem?

$$A :\equiv \forall x^X \exists y^Y \forall z^Z P(x, y, z)$$

In general we cannot hope to produce a direct computable witness for $\exists y$. But suppose we double negate and Skolemize:

What is the computational meaning of a Π_3-theorem?

$$A :\equiv \forall x^X \exists y^Y \forall z^Z P(x, y, z)$$

In general we cannot hope to produce a direct computable witness for $\exists y$. But suppose we double negate and Skolemize:

$$\begin{aligned} \neg A &\leftrightarrow \exists x \forall y \exists z \neg P(x, y, z) \\ &\leftrightarrow \exists x, \xi^{Y \to Z} \forall y \neg P(x, y, \xi(y)) \end{aligned}$$

What is the computational meaning of a Π_3-theorem?

$$A :\equiv \forall x^X \exists y^Y \forall z^Z P(x, y, z)$$

In general we cannot hope to produce a direct computable witness for $\exists y$. But suppose we double negate and Skolemize:

$$\begin{aligned}
\neg A &\leftrightarrow \exists x \forall y \exists z \neg P(x, y, z) \\
&\leftrightarrow \exists x, \xi^{Y \to Z} \forall y \neg P(x, y, \xi(y)) \\
\neg\neg A &\leftrightarrow \forall x, \xi \exists y \neg\neg P(x, y, \xi(y)) \\
&\leftrightarrow \forall x, \xi \exists y \; P(x, y, \xi(y))
\end{aligned}$$

What is the computational meaning of a Π_3-theorem?

$$A :\equiv \forall x^X \exists y^Y \forall z^Z P(x, y, z)$$

In general we cannot hope to produce a direct computable witness for $\exists y$. But suppose we double negate and Skolemize:

$$\begin{aligned}
\neg A &\leftrightarrow \exists x \forall y \exists z \neg P(x, y, z) \\
&\leftrightarrow \exists x, \xi^{Y \to Z} \forall y \neg P(x, y, \xi(y)) \\
\neg\neg A &\leftrightarrow \forall x, \xi \exists y \neg\neg P(x, y, \xi(y)) \\
&\leftrightarrow \forall x, \xi \exists y \; P(x, y, \xi(y))
\end{aligned}$$

Over classical logic

$$\underbrace{\forall x \exists y \forall z P(x, y, z)}_{y \text{ ideal (for all } z)} \leftrightarrow \underbrace{\forall x, \xi \exists y P(x, y, \xi(y))}_{y \text{ approximate relative to } \xi}$$

but we can realize the R.H.S.

Let $f : \mathbb{N} \to \mathbb{N}$ be an arbitrary function.

INFINITARY. For any $x \in \mathbb{N}$ there exists some $y \geq x$ such that

$$\forall z[z \geq x \to f(y) \leq f(z)].$$

Let $f\colon \mathbb{N} \to \mathbb{N}$ be an arbitrary function.

INFINITARY. For any $x \in \mathbb{N}$ there exists some $y \geq x$ such that

$$\forall z[z \geq x \to f(y) \leq f(z)].$$

FINITARY. For any $x \in \mathbb{N}$ and $\xi\colon \mathbb{N} \to \mathbb{N}$ there exists some $y \geq x$ such that

$$\xi(y) \geq y \to f(y) \leq f(\xi(y)).$$

Let $f\colon \mathbb{N} \to \mathbb{N}$ be an arbitrary function.

INFINITARY. For any $x \in \mathbb{N}$ there exists some $y \geq x$ such that

$$\forall z[z \geq x \to f(y) \leq f(z)].$$

FINITARY. For any $x \in \mathbb{N}$ and $\xi\colon \mathbb{N} \to \mathbb{N}$ there exists some $y \geq x$ such that

$$\xi(y) \geq y \to f(y) \leq f(\xi(y)).$$

We can compute y by *learning*, as follows:

$$y := \begin{cases} x & \text{if } \xi(x) \geq x \to f(x) \leq f(\xi(x)) \\ \xi(x) & \text{if } \xi^{(2)}(x) \geq \xi(x) \to f(\xi(x)) \leq f(\xi^{(2)}(x)) \\ \xi^{(2)}(x) & \text{if } \xi^{(3)}(x) \geq \xi^{(2)}(x) \to f(\xi^{(2)}(x)) \leq f(\xi^{(3)}(x)) \\ \dots & \dots \end{cases}$$

Terminates since otherwise we'd have $f(x) > f(\xi(x)) > f(\xi^{(2)}(x)) > \dots$

A more interesting example: Cauchy convergence of a monotone sequence $(a_i) \in [0,1]^\omega$

INFINITARY. For any x there exists y such that $i, j \geq y$ implies $|a_i - a_j| < 2^{-x}$.

FINITARY (T. TAO). For any x and $\xi \colon \mathbb{N} \to \mathbb{N}$ there exists Y such that any $0 \leq a_0 \leq \ldots \leq a_Y \leq 1$ there exists y with $0 \leq y < y + \xi(y) \leq Y$ such that $|a_i - a_j| < 2^{-x}$ for all $y \leq i, j \leq \xi(y)$.

Moreover, we can show that $Y \leq (\lambda i.i + \xi(i))^{(2^x)}(0)$.

The general problem of constructing a realizer for a finitary formulation of a classical theorem is highly non-trivial. We can appeal to Gödel's functional interpretation (more precisely his ND interpretation):

$$A \rightsquigarrow A^{\neg\neg} \rightsquigarrow \exists x \forall y |A^{\neg\neg}|_y^x$$

which allows us to extract from a proof $\mathcal{P}$ of A a term t satisfying $\forall y |A^{\neg\neg}|_y^t$

The general problem of constructing a realizer for a finitary formulation of a classical theorem is highly non-trivial. We can appeal to Gödel's functional interpretation (more precisely his ND interpretation):

$$A \rightsquigarrow A^{\neg\neg} \rightsquigarrow \exists x \forall y |A^{\neg\neg}|_y^x$$

which allows us to extract from a proof $\mathcal{P}$ of A a term t satisfying $\forall y |A^{\neg\neg}|_y^t$

However, a brute force extraction can yield terms which are highly inefficient and difficult to understand.

The general problem of constructing a realizer for a finitary formulation of a classical theorem is highly non-trivial. We can appeal to Gödel's functional interpretation (more precisely his ND interpretation):

$$A \rightsquigarrow A^{\neg\neg} \rightsquigarrow \exists x \forall y |A^{\neg\neg}|_y^x$$

which allows us to extract from a proof $\mathcal{P}$ of A a term t satisfying $\forall y |A^{\neg\neg}|_y^t$

However, a brute force extraction can yield terms which are highly inefficient and difficult to understand.

Given the wide range of applications of the functional interpretation in modern proof theory, it can be extremely useful to devise refinements of the functional interpretation which help us extract better terms.

The general problem of constructing a realizer for a finitary formulation of a classical theorem is highly non-trivial. We can appeal to Gödel's functional interpretation (more precisely his ND interpretation):

$$A \rightsquigarrow A^{\neg\neg} \rightsquigarrow \exists x \forall y |A^{\neg\neg}|_y^x$$

which allows us to extract from a proof $\mathcal{P}$ of A a term t satisfying $\forall y |A^{\neg\neg}|_y^t$

However, a brute force extraction can yield terms which are highly inefficient and difficult to understand.

Given the wide range of applications of the functional interpretation in modern proof theory, it can be extremely useful to devise refinements of the functional interpretation which help us extract better terms.

My paper presents one such refinement, which enriches the usual interpreting system of higher type recursors with *learning algorithms* that produce realizing terms which are more efficient and intuitive.

The intuition behind learning algorithms

Suppose we have a decidable predicate G on X, and we want to find some $x \in X$ satisfying $\mathsf{G}(x)$.

The intuition behind learning algorithms

Suppose we have a decidable predicate G on X, and we want to find some $x \in X$ satisfying $\mathsf{G}(x)$.

Let's take some arbitrary x_0. If $\mathsf{G}(x_0)$ holds then we're done. On the other hand, if $\neg\mathsf{G}(x_0)$ and we fail we often learn a useful piece of constructive information $\xi(x_0)$.

The intuition behind learning algorithms

Suppose we have a decidable predicate G on X, and we want to find some $x \in X$ satisfying $\mathsf{G}(x)$.

Let's take some arbitrary x_0. If $\mathsf{G}(x_0)$ holds then we're done. On the other hand, if $\neg\mathsf{G}(x_0)$ and we fail we often learn a useful piece of constructive information $\xi(x_0)$.

We can then use this to update our original guess with a better one $x_1 := x_0 \oplus \xi(x_0)$, and continue:

The intuition behind learning algorithms

Suppose we have a decidable predicate G on X, and we want to find some $x \in X$ satisfying $\mathsf{G}(x)$.

Let's take some arbitrary x_0. If $\mathsf{G}(x_0)$ holds then we're done. On the other hand, if $\neg\mathsf{G}(x_0)$ and we fail we often learn a useful piece of constructive information $\xi(x_0)$.

We can then use this to update our original guess with a better one $x_1 := x_0 \oplus \xi(x_0)$, and continue:

$$x := \begin{cases} x_0 & \text{if } \mathsf{G}(x_0) \\ x_1 := x_0 \oplus \xi(x_0) & \text{if } \mathsf{G}(x_1) \\ x_2 := x_1 \oplus \xi(x_1) & \text{if } \mathsf{G}(x_2) \\ \dots & \dots \end{cases}$$

The idea is that we eventually reach some x_k satisfying $\mathsf{G}(x_k)$.

Learning algorithms - a formal definition

A learning algorithm of type X, L is a tuple $\mathcal{L} = (\mathsf{G}, \xi, \oplus)$ where

- $\mathsf{G} : X \to \mathbb{B}$ is a decidable predicate which tests whether an element $x \in X$ is 'good';
- $\xi : X \to L$ and $\oplus \colon X \times L \to X$ are responsible for learning, and will be used to map bad objects $x \in X$ to improvements $x \oplus \xi(x)$;

Learning algorithms - a formal definition

A learning algorithm of type X, L is a tuple $\mathcal{L} = (\mathsf{G}, \xi, \oplus)$ where

- $\mathsf{G} : X \to \mathbb{B}$ is a decidable predicate which tests whether an element $x \in X$ is 'good';
- $\xi : X \to L$ and $\oplus \colon X \times L \to X$ are responsible for learning, and will be used to map bad objects $x \in X$ to improvements $x \oplus \xi(x)$;

The learning procedure $\mathcal{L}[x]$ starting at $x \in X$ is a sequence $(x_i) \in X^{\mathbb{N}}$ defined by

$$x_0 := x \qquad \text{and} \qquad x_{i+1} := \begin{cases} x_i & \text{if } \mathsf{G}(x_i) \\ x_i \oplus \xi(x_i) & \text{if } \neg\mathsf{G}(x_i) \end{cases}$$

Learning algorithms - a formal definition

A learning algorithm of type X, L is a tuple $\mathcal{L} = (\mathsf{G}, \xi, \oplus)$ where

- $\mathsf{G} : X \to \mathbb{B}$ is a decidable predicate which tests whether an element $x \in X$ is 'good';
- $\xi : X \to L$ and $\oplus \colon X \times L \to X$ are responsible for learning, and will be used to map bad objects $x \in X$ to improvements $x \oplus \xi(x)$;

The learning procedure $\mathcal{L}[x]$ starting at $x \in X$ is a sequence $(x_i) \in X^{\mathbb{N}}$ defined by

$$x_0 := x \qquad \text{and} \qquad x_{i+1} := \begin{cases} x_i & \text{if } \mathsf{G}(x_i) \\ x_i \oplus \xi(x_i) & \text{if } \neg\mathsf{G}(x_i) \end{cases}$$

The limit of $\mathcal{L}[x]$ is defined as

$$\lim \mathcal{L}[x] := x_k$$

where x_k is the least point satisfying $\mathsf{G}(x_k)$ (whenever it exists).

ROUGH IDEA. Suppose that $\exists x A(x)$ is a classical theorem, and

$$\forall \xi \exists x A'_\xi(x, \xi)$$

a finitization of A. Then x can be computed in the limit of a learning procedure parametrised by ξ:

$$x := F(\lim \mathcal{L}_\xi[x_0])$$

for some x_0.

ROUGH IDEA. Suppose that $\exists x A(x)$ is a classical theorem, and

$$\forall \xi \exists x A'_\xi(x, \xi)$$

a finitization of A. Then x can be computed in the limit of a learning procedure parametrised by ξ:

$$x := F(\lim \mathcal{L}_\xi[x_0])$$

for some x_0.

This idea is made precise in the paper, where a collection of concrete results of this kind are given, relating Gödel's functional interpretation of induction and comprehension principles to learning procedures.

In the remainder of the talk, however, I will simply give some illustrations.

EXAMPLE 1: The quantifier-free minimum principle

$$\mathsf{QFMin}: \ \exists x P(x) \to \exists y (P(y) \wedge \forall z \prec y \neg P(z)).$$

EXAMPLE 1: The quantifier-free minimum principle

$$\mathsf{QFMin}: \ \exists x P(x) \to \exists y (P(y) \wedge \forall z \prec y \neg P(z)).$$

This has an ND interpretation given by

$$(*) \ \ \forall x, \xi \exists y (P(x) \to P(y) \wedge (\xi(y) \prec y \to \neg P(\xi(y))))$$

"For all x, ξ there exists some y such that whenever $P(x)$ holds then $P(y)$ holds and y is approximately minimal with respect to $\xi(y)$"

EXAMPLE 1: The quantifier-free minimum principle

$$\mathsf{QFMin}: \ \exists x P(x) \to \exists y (P(y) \wedge \forall z \prec y \neg P(z)).$$

This has an ND interpretation given by

$$(*) \ \ \forall x, \xi \exists y (P(x) \to P(y) \wedge (\xi(y) \prec y \to \neg P(\xi(y))))$$

"For all x, ξ there exists some y such that whenever $P(x)$ holds then $P(y)$ holds and y is approximately minimal with respect to $\xi(y)$"

We can compute y in x and ξ using the following idea

$$y := \begin{cases} x & \text{if } \xi(x) \prec x \to \neg P(\xi(x)) \\ \xi(x) & \text{if } \xi^{(2)}(x) \prec \xi(x) \to \neg P(\xi^{(2)}(x)) \\ \xi^{(2)}(x) & \text{if } \dots \\ \dots & \dots \end{cases}$$

THEOREM. Define $\mathcal{L}_\xi := (\mathsf{G}_\xi, \xi, \pi_2)$ where

$$\mathsf{G}_\xi(x) \leftrightarrow [\xi(x) \prec x \to \neg P(\xi(x))].$$

Then the ND interpretation of QFMin, given by

$$\forall x, \xi \exists y (P(x) \to P(y) \wedge (\xi(y) \prec y \to \neg P(\xi(y))))$$

is realized by

$$\lambda x, \xi \,.\, \lim \mathcal{L}_\xi[x].$$

REMARK. A general result dealing with well-founded induction for arbitrary formulas and relations $\prec$ is given in the paper.

Example 2, following (Schwichtenberg 2008). For any two natural numbers $a, b > 0$ there exist integers m, n such that $am + bn \mid a, b$.

Classical proof. Use a variant of QFMin relative to the ordering $(x, y) \prec (x', y') := ax + by < ax' + by'$.

Example 2, following (Schwichtenberg 2008). For any two natural numbers $a, b > 0$ there exist integers m, n such that $am + bn \mid a, b$.

Classical proof. Use a variant of QFMin relative to the ordering $(x, y) \prec (x', y') := ax + by < ax' + by'$.

A program for computing m, n in a, b can be extracted, namely a learning procedure of type $(\mathbb{N}^{(2)})^*$, $\mathbb{N}^{(2)}$ given by

$$(m, n) := \mathsf{tail}(\lim \mathcal{L}_{a,b}[\langle e_0, e_1 \rangle])$$

where $\mathcal{L}_{a,b} = (\mathsf{G}_{a,b}, \xi_{a,b}, ::)$ for

- $\mathsf{G}_{a,b}(s) \leftrightarrow \mathrm{rem}(s_{l-2} \cdot (a, b), s_{l-1} \cdot (a, b)) = 0$
- $\xi_{a,b}(s) \leftrightarrow s_{l-2} - \mathrm{quot}(s_{l-2} \cdot (a, b), s_{l-1} \cdot (a, b)) s_{l-1}$
- $s :: x := \langle s_0, \ldots, s_{l-1}, x \rangle$.

Example 2, following (Schwichtenberg 2008). For any two natural numbers $a, b > 0$ there exist integers m, n such that $am + bn \mid a, b$.

Classical proof. Use a variant of QFMin relative to the ordering $(x, y) \prec (x', y') := ax + by < ax' + by'$.

A program for computing m, n in a, b can be extracted, namely a learning procedure of type $(\mathbb{N}^{(2)})^*$, $\mathbb{N}^{(2)}$ given by

$$(m, n) := \mathsf{tail}(\lim \mathcal{L}_{a,b}[\langle e_0, e_1 \rangle])$$

where $\mathcal{L}_{a,b} = (\mathsf{G}_{a,b}, \xi_{a,b}, ::)$ for

- $\mathsf{G}_{a,b}(s) \leftrightarrow \mathrm{rem}(s_{l-2} \cdot (a, b), s_{l-1} \cdot (a, b)) = 0$
- $\xi_{a,b}(s) \leftrightarrow s_{l-2} - \mathrm{quot}(s_{l-2} \cdot (a, b), s_{l-1} \cdot (a, b)) s_{l-1}$
- $s :: x := \langle s_0, \ldots, s_{l-1}, x \rangle$.

This is just the Euclidean algorithm!

Meta-example: The double negation shift

$$\mathsf{DNS}: \ \forall n \neg\neg \exists x^X \forall y P_n(x,y) \to \neg\neg \forall n \exists x \forall y P_n(x,y)$$

Meta-example: The double negation shift

$$\mathsf{DNS}: \ \forall n \neg\neg \exists x^X \forall y P_n(x,y) \to \neg\neg \forall n \exists x \forall y P_n(x,y)$$

This has a (partial) functional interpretation given by

$$\underbrace{\forall n, \xi \exists x P_n(x, \xi(x))} \to \underbrace{\forall \omega, \varphi \exists \alpha^{X^{\mathbb{N}}} P_{\omega\alpha}(\alpha(\omega\alpha), \varphi\alpha)}$$

"If, for each n, P_n is approximately witnessed by some $x \in X$ relative to ξ, then we can produce a 'global' witness $\alpha \in X^{\mathbb{N}}$ for the conclusion which is approximately correct relative to $\varphi\alpha$ at point $\omega\alpha$"

Meta-example: The double negation shift

$$\mathsf{DNS}: \ \forall n \neg\neg \exists x^X \forall y P_n(x,y) \to \neg\neg \forall n \exists x \forall y P_n(x,y)$$

This has a (partial) functional interpretation given by

$$\underbrace{\forall n, \xi \exists x P_n(x, \xi(x))}_{(\mathcal{L}_{n,\xi})_{n<\infty}} \to \underbrace{\forall \omega, \varphi \exists \alpha^{X^{\mathbb{N}}} P_{\omega\alpha}(\alpha(\omega\alpha), \varphi\alpha)}_{\mathcal{L}_{\infty,(\omega,\varphi)}}$$

"If, for each n, P_n is approximately witnessed by some $x \in X$ relative to ξ, then we can produce a 'global' witness $\alpha \in X^{\mathbb{N}}$ for the conclusion which is approximately correct relative to $\varphi\alpha$ at point $\omega\alpha$"

Is there some sort of formal construction from a collection of 'pointwise' learning algorithms to a 'global' learning algorithm:

$$(\mathcal{L}_{n,\xi})_{n<\infty} \mapsto \mathcal{L}_{\infty,(\omega,\phi)}?$$

Meta-example: The double negation shift

$$\mathsf{DNS}: \ \forall n \neg\neg \exists x^X \forall y P_n(x,y) \to \neg\neg \forall n \exists x \forall y P_n(x,y)$$

This has a (partial) functional interpretation given by

$$\underbrace{\forall n, \xi \exists x P_n(x, \xi(x))}_{(\mathcal{L}_{n,\xi})_{n<\infty}} \to \underbrace{\forall \omega, \varphi \exists \alpha^{X^{\mathbb{N}}} P_{\omega\alpha}(\alpha(\omega\alpha), \varphi\alpha)}_{\mathcal{L}_{\infty,(\omega,\varphi)}}$$

"If, for each n, P_n is approximately witnessed by some $x \in X$ relative to ξ, then we can produce a 'global' witness $\alpha \in X^{\mathbb{N}}$ for the conclusion which is approximately correct relative to $\varphi\alpha$ at point $\omega\alpha$"

Is there some sort of formal construction from a collection of 'pointwise' learning algorithms to a 'global' learning algorithm:

$$(\mathcal{L}_{n,\xi})_{n<\infty} \mapsto \mathcal{L}_{\infty,(\omega,\phi)}?$$

Yes! But I will just give an illustration here.

Example 3: Law of excluded middle for Σ^0_1-formulas

$$\forall n \exists b^{\mathbb{B}} (\exists x Q_n(x) \vee_b \forall y \neg Q_n(y))$$
$$\rightsquigarrow \forall n \exists b, x \forall y (Q_n(x) \vee_b \neg Q_n(y))$$

Example 3: Law of excluded middle for Σ^0_1-formulas

$$\forall n \exists b^{\mathbb{B}} (\exists x Q_n(x) \vee_b \forall y \neg Q_n(y))$$
$$\rightsquigarrow \forall n \exists b, x \forall y (Q_n(x) \vee_b \neg Q_n(y))$$

This has a functional interpretation given by

$$\forall n, \xi \exists b, x (Q_n(x) \vee_b \neg Q_n(\xi(b, x)))$$

Example 3: Law of excluded middle for Σ^0_1-formulas

$$\forall n \exists b^{\mathbb{B}} (\exists x Q_n(x) \vee_b \forall y \neg Q_n(y))$$
$$\rightsquigarrow \forall n \exists b, x \forall y (Q_n(x) \vee_b \neg Q_n(y))$$

This has a functional interpretation given by

$$\forall n, \xi \exists b, x (Q_n(x) \vee_b \neg Q_n(\xi(b, x)))$$

which is realized by a learning procedure of length at most two:

$$b, x = \begin{cases} \bot, 0 & \text{if } \neg Q_n(\xi(\bot, 0)) \\ \top, \xi(\bot, 0) & \text{otherwise.} \end{cases}$$

Example 4: Σ^0_1-arithmetic comprehension

$$\exists \alpha^{\mathbb{N}\to\mathbb{B}\times\mathbb{N}} \forall n \forall y [Q_n(\alpha(n)_1) \vee_{\alpha(n)_0} \neg Q_n(y)]$$

Example 4: Σ^0_1-arithmetic comprehension

$$\exists \alpha^{\mathbb{N}\to\mathbb{B}\times\mathbb{N}} \forall n \forall y [Q_n(\alpha(n)_1) \vee_{\alpha(n)_0} \neg Q_n(y)]$$

i.e. $\alpha(n)_0 \in \mathbb{B}$ indicates the truth of $\exists x Q_n(x)$ and $\alpha(n)_1 \in \mathbb{N}$ provides a witness. This has a functional interpretation given by

$$\forall \omega, \varphi \exists \alpha [Q_{\omega\alpha}(\alpha(\omega\alpha)_1) \vee_{\alpha(\omega\alpha)_0} \neg Q_{\omega\alpha}(\varphi\alpha)]$$

Example 4: Σ^0_1-arithmetic comprehension

$$\exists \alpha^{\mathbb{N}\to\mathbb{B}\times\mathbb{N}} \forall n \forall y [Q_n(\alpha(n)_1) \vee_{\alpha(n)_0} \neg Q_n(y)]$$

i.e. $\alpha(n)_0 \in \mathbb{B}$ indicates the truth of $\exists x Q_n(x)$ and $\alpha(n)_1 \in \mathbb{N}$ provides a witness. This has a functional interpretation given by

$$\forall \omega, \varphi \exists \alpha [Q_{\omega\alpha}(\alpha(\omega\alpha)_1) \vee_{\alpha(\omega\alpha)_0} \neg Q_{\omega\alpha}(\varphi\alpha)]$$

which can be realized by a learning procedure on $\mathbb{N} \to \mathbb{B} \times \mathbb{N}$ of unbounded length, but of 'pointwise' length at most two.

$$\alpha = \begin{cases} \underbrace{[\,]}_{\alpha_0} := \lambda n.(\bot, 0) & \text{if } \neg Q_{\omega\alpha_0}(\varphi\alpha_0) \\ \underbrace{[\omega\alpha_0 \mapsto (\top, \varphi\alpha_0)]}_{\alpha_1} & \text{if } \alpha_1(\omega\alpha_1)_0 = \bot \to \neg Q_{\omega\alpha_1}(\varphi\alpha_1) \\ \underbrace{[\omega\alpha_0 \mapsto (\top, \varphi\alpha_0), \omega\alpha_1 \mapsto (\top, \varphi\alpha_1)]}_{\alpha_2} & \text{if } \alpha_2(\omega\alpha_2)_0 = \bot \to \neg Q_{\omega\alpha_2}(\varphi\alpha_2) \\ \dots & \dots \end{cases}$$

SUMMARY. For the instance of DNS given by

$$\forall n \neg\neg \exists b, x \forall y [Q_n(x) \vee_b \neg Q_n(y)] \to \neg\neg \forall n \exists b, x \forall y [Q_n(x) \vee_b Q_n(y)]$$

we have $(\mathcal{L}_{n,\xi})_{n<\infty}$ given by

$$b, x = \begin{cases} \bot, 0 & \text{if } \neg Q_n(\xi(\bot, 0)) \\ \top, \xi(\bot, 0) & \text{otherwise.} \end{cases}$$

and $\mathcal{L}_{\infty,(\omega,\varphi)}$ given by

$$\alpha = \begin{cases} [\,] = \lambda n.(\bot, 0) =: \alpha_0 & \text{if } \neg Q_{\omega\alpha_0}(\varphi\alpha_0) \\ [\omega\alpha_0 \mapsto (\top, \varphi\alpha_0)] =: \alpha_1 & \text{if } \alpha_1(\omega\alpha_1)_0 = \bot \to \neg Q_{\omega\alpha_1}(\varphi\alpha_1) \\ [\omega\alpha_0 \mapsto (\top, \varphi\alpha_0), \omega\alpha_1 \mapsto (\top, \varphi\alpha_1)] & \text{if } \ldots \\ \ldots & \ldots \end{cases}$$

Directions for the future

- To establish new complexity results on the length of learning procedures extracted from proofs.

- Try to link our approach to other computational interpretations of classical logic based on learning e.g. the Aschieri-Berardi learning realizability (Aschieri-Berardi 2010).

- Formalise everything so that learning procedures can be automatically extracted, and resulting programs decorated with information indicating how they behave.

- Extend the idea to stronger subsystems of mathematics. I conjecture that weak forms of Zorn's lemma of the kind used to prove Kruskal's theorem can be interpreted via a learning procedure which computes approximations to maximal elements in chain complete posets.