Learning, Loops and Limits

Thomas Powell

Technische Universität Darmstadt

Logic and Theory Group Seminar

Universität Bern, Switzerland

27 October 2016

A REASONABLY LONG INTRODUCTION...

Let's begin by looking at a simple non-constructive theorem, sometimes called the 'drinkers paradox':

Let's begin by looking at a simple non-constructive theorem, sometimes called the 'drinkers paradox':

$$\exists n \in \mathbb{N} \forall m \in \mathbb{N} (P_m \to P_n)$$

or... "In a pub there is always a person such that if anyone else is drinking, then that person in drinking"

Let's begin by looking at a simple non-constructive theorem, sometimes called the 'drinkers paradox':

$$\exists n \in \mathbb{N} \forall m \in \mathbb{N} (P_m \to P_n)$$

or... "In a pub there is always a person such that if anyone else is drinking, then that person in drinking"

The drinkers paradox has a quick proof using classical logic:

$$\dfrac{\exists k P_k \vee \forall k \neg P_k \qquad \dfrac{\dfrac{P_k \to \forall m (P_m \to P_k)}{P_k \to \exists n \forall m (P_m \to P_n)}}{\exists k P_k \to \exists n \forall m (P_m \to P_n)} \qquad \dfrac{\dfrac{\forall k \neg P_k \to P_m \to P_0}{\forall k \neg P_k \to \forall m (P_m \to P_0)}}{\forall k \neg P_k \to \exists n \forall m (P_m \to P_n)}}{\exists n \forall m (P_m \to P_n)}$$

Let's begin by looking at a simple non-constructive theorem, sometimes called the 'drinkers paradox':

$$\exists n \in \mathbb{N} \forall m \in \mathbb{N}(P_m \to P_n)$$

or... "In a pub there is always a person such that if anyone else is drinking, then that person in drinking"

The drinkers paradox has a quick proof using classical logic:

$$\dfrac{\exists k P_k \vee \forall k \neg P_k \qquad \dfrac{\dfrac{P_k \to \forall m(P_m \to P_k)}{P_k \to \exists n \forall m(P_m \to P_n)}}{\exists k P_k \to \exists n \forall m(P_m \to P_n)} \qquad \dfrac{\dfrac{\forall k \neg P_k \to P_m \to P_0}{\forall k \neg P_k \to \forall m(P_m \to P_0)}}{\forall k \neg P_k \to \exists n \forall m(P_m \to P_n)}}{\exists n \forall m(P_m \to P_n)}$$

However, in general there is no *effective* way of realizing $\exists n$.

Q. What is the constructive interpretation of this theorem?

Old fashioned method: Hilbert's ϵ-calculus

Idea: Replace quantifiers by 'ideal' ϵ-terms:

$$\exists k A(k) \rightsquigarrow A(\epsilon_k A),$$

and quantifier axioms by critical formulas:

$$A(t) \to A(\epsilon_k A).$$

Old fashioned method: Hilbert's ϵ-calculus

Idea: Replace quantifiers by 'ideal' ϵ-terms:

$$\exists k A(k) \rightsquigarrow A(\epsilon_k A),$$

and quantifier axioms by critical formulas:

$$A(t) \to A(\epsilon_k A).$$

1. *Translation.* Convert proofs in predicate logic to proofs in the epsilon calculus. Instances of quantifier axioms are replaced by critical formulas.

2. *Epsilon elimination (roughly!).* Suppose we only use a finite set of critical formulas. Interpret all ϵ-terms by 0. If we find a mistake i.e. $A(t) \wedge \neg A(0)$, we 'learn' from this mistake and update $\epsilon_k A \mapsto t$.

Interpreted proof:

$$\dfrac{\exists k P_k \lor \forall k \neg P_k \qquad \dfrac{\dfrac{P_k \to \forall m (P_m \to P_k)}{P_k \to \exists n \forall m (P_m \to P_n)}}{\exists k P_k \to \exists n \forall m (P_m \to P_n)} \qquad \dfrac{\dfrac{\dfrac{\neg P_m \to P_m \to P_0}{\forall k \neg P_k \to P_m \to P_0}}{\forall k \neg P_k \to \forall m (P_m \to P_0)}}{\forall k \neg P_k \to \exists n \forall m (P_m \to P_n)}}{\exists n \forall m (P_m \to P_n)}$$

Critical formulas:

ϵ-elimination:

Interpreted proof:

$$
\dfrac{\boxed{\exists k P_k \vee \forall k \neg P_k} \qquad \dfrac{\dfrac{P_k \to \forall m (P_m \to P_k)}{P_k \to \exists n \forall m (P_m \to P_n)}}{\exists k P_k \to \exists n \forall m (P_m \to P_n)} \qquad \dfrac{\dfrac{\dfrac{\neg P_m \to P_m \to P_0}{\forall k \neg P_k \to P_m \to P_0}\ \boxed{\forall \mathrm{ax}}}{\forall k \neg P_k \to \forall m (P_m \to P_0)}}{\forall k \neg P_k \to \exists n \forall m (P_m \to P_n)}}{\exists n \forall m (P_m \to P_n)}
$$

Critical formulas:

ϵ-elimination:

Interpreted proof:

$$\dfrac{\boxed{P_{\epsilon_k} \vee \neg P_{\epsilon_k}} \qquad \dfrac{P_{\epsilon_k} \to \forall m(P_m \to P_{\epsilon_k})}{P_{\epsilon_k} \to \exists n \forall m(P_m \to P_n)} \qquad \dfrac{\dfrac{\dfrac{\neg P_m \to P_m \to P_0}{\neg P_{\epsilon_k} \to P_m \to P_0}\ \boxed{\forall\text{ax}}}{\neg P_{\epsilon_k} \to \forall m(P_m \to P_0)}}{\neg P_{\epsilon_k} \to \exists n \forall m(P_m \to P_n)}}{\exists n \forall m(P_m \to P_n)}$$

Critical formulas:

$$P_m \to P_{\epsilon_k}$$

ϵ-elimination:

Interpreted proof:

$$\dfrac{P_{\epsilon_k} \vee \neg P_{\epsilon_k} \qquad \dfrac{P_{\epsilon_k} \to \forall m(P_m \to P_{\epsilon_k})}{P_{\epsilon_k} \to \exists n \forall m(P_m \to P_n)} \qquad \dfrac{\dfrac{\dfrac{\neg P_m \to P_m \to P_0}{\neg P_{\epsilon_k} \to P_m \to P_0}}{\neg P_{\epsilon_k} \to \forall m(P_m \to P_0)}}{\neg P_{\epsilon_k} \to \exists n \forall m(P_m \to P_n)}}{\exists n \boxed{\forall m(P_m \to P_n)}}$$

Critical formulas:

$$P_m \to P_{\epsilon_k}$$

ϵ-elimination:

Interpreted proof:

$$\dfrac{P_{\epsilon_k} \vee \neg P_{\epsilon_k} \qquad \dfrac{P_{\epsilon_k} \to P_{\epsilon_m \epsilon_k} \to P_{\epsilon_k}}{P_{\epsilon_k} \to \exists n (P_{\epsilon_m n} \to P_n)} \qquad \dfrac{\dfrac{\neg P_{\varepsilon_m 0} \to P_{\varepsilon_m 0} \to P_0}{\neg P_{\epsilon_k} \to P_{\epsilon_m 0} \to P_0}}{\neg P_{\epsilon_k} \to \exists n (P_{\epsilon_m n} \to P_n)}}{\exists n \boxed{(P_{\epsilon_m n} \to P_n)}}$$

Critical formulas:

$$P_{\epsilon_m 0} \to P_{\epsilon_k}$$

ϵ-elimination:

Interpreted proof:

$$\dfrac{P_{\epsilon_k} \vee \neg P_{\epsilon_k} \qquad \dfrac{P_{\epsilon_k} \to P_{\epsilon_m \epsilon_k} \to P_{\epsilon_k}}{P_{\epsilon_k} \to \exists n (P_{\epsilon_m n} \to P_n)}\ \boxed{\exists\text{ax}} \qquad \dfrac{\dfrac{\neg P_{\varepsilon_m 0} \to P_{\varepsilon_m 0} \to P_0}{\neg P_{\epsilon_k} \to P_{\epsilon_m 0} \to P_0}}{\neg P_{\epsilon_k} \to \exists n(P_{\epsilon_m n} \to P_n)}\ \boxed{\exists\text{ax}}}{\boxed{\exists n(P_{\epsilon_m n} \to P_n)}}$$

Critical formulas:

$$P_{\epsilon_m 0} \to P_{\epsilon_k}$$

ϵ-elimination:

Interpreted proof:

$$\dfrac{P_{\epsilon_k} \vee \neg P_{\epsilon_k} \qquad \dfrac{P_{\epsilon_k} \to P_{\epsilon_m \epsilon_k} \to P_{\epsilon_k}}{P_{\epsilon_k} \to (P_{\epsilon_m \epsilon_n} \to P_{\epsilon_n})}\ \boxed{\exists\text{ax}} \qquad \dfrac{\dfrac{\neg P_{\varepsilon_m 0} \to P_{\varepsilon_m 0} \to P_0}{\neg P_{\epsilon_k} \to P_{\epsilon_m 0} \to P_0}}{\neg P_{\epsilon_k} \to (P_{\epsilon_m \epsilon_n} \to P_{\epsilon_n})}\ \boxed{\exists\text{ax}}}{\boxed{P_{\epsilon_m \epsilon_n} \to P_{\epsilon_n}}}$$

Critical formulas:

$$P_{\epsilon_m 0} \to P_{\epsilon_k}$$
$$(P_{\epsilon_m \epsilon_k} \to P_{\epsilon_k}) \to (P_{\epsilon_m \epsilon_n} \to P_{\epsilon_n})$$
$$(P_{\epsilon_m 0} \to P_0) \to (P_{\epsilon_m \epsilon_n} \to P_{\epsilon_n})$$

ϵ-elimination:

Interpreted proof:

$$\dfrac{P_{\epsilon_k} \vee \neg P_{\epsilon_k} \qquad \dfrac{P_{\epsilon_k} \to P_{\epsilon_m \epsilon_k} \to P_{\epsilon_k}}{P_{\epsilon_k} \to (P_{\epsilon_m \epsilon_n} \to P_{\epsilon_n})} \qquad \dfrac{\dfrac{\neg P_{\varepsilon_m 0} \to P_{\varepsilon_m 0} \to P_0}{\neg P_{\epsilon_k} \to P_{\epsilon_m 0} \to P_0}}{\neg P_{\epsilon_k} \to (P_{\epsilon_m \epsilon_n} \to P_{\epsilon_n})}}{P_{\epsilon_m \epsilon_n} \to P_{\epsilon_n}}$$

Critical formulas:

$$P_{\epsilon_m 0} \to P_{\epsilon_k}$$
$$(P_{\epsilon_m \epsilon_k} \to P_{\epsilon_k}) \to (P_{\epsilon_m \epsilon_n} \to P_{\epsilon_n})$$
$$(P_{\epsilon_m 0} \to P_0) \to (P_{\epsilon_m \epsilon_n} \to P_{\epsilon_n})$$

ϵ-elimination:

Interpreted proof:

$$\dfrac{P_{\epsilon_k} \vee \neg P_{\epsilon_k} \qquad \dfrac{P_{\epsilon_k} \to P_{\epsilon_m \epsilon_k} \to P_{\epsilon_k}}{P_{\epsilon_k} \to (P_{\epsilon_m \epsilon_n} \to P_{\epsilon_n})} \qquad \dfrac{\dfrac{\neg P_{\varepsilon_m 0} \to P_{\varepsilon_m 0} \to P_0}{\neg P_{\epsilon_k} \to P_{\epsilon_m 0} \to P_0}}{\neg P_{\epsilon_k} \to (P_{\epsilon_m \epsilon_n} \to P_{\epsilon_n})}}{P_{\epsilon_m \epsilon_n} \to P_{\epsilon_n}}$$

Critical formulas:

$$\begin{array}{cc} P_{\epsilon_m 0} \to P_0 & ? \\ (P_{\epsilon_m 0} \to P_0) \to (P_{\epsilon_m 0} \to P_0) & \checkmark \\ (P_{\epsilon_m 0} \to P_0) \to (P_{\epsilon_m 0} \to P_0) & \checkmark \end{array}$$

ϵ-elimination:

- Try $\epsilon_k = \epsilon_n = 0$. Works unless $P_{\epsilon_m 0} \wedge \neg P_0$.

Interpreted proof:

$$\dfrac{P_{\epsilon_k} \vee \neg P_{\epsilon_k} \qquad \dfrac{P_{\epsilon_k} \to P_{\epsilon_m \epsilon_k} \to P_{\epsilon_k}}{P_{\epsilon_k} \to (P_{\epsilon_m \epsilon_n} \to P_{\epsilon_n})} \qquad \dfrac{\dfrac{\neg P_{\varepsilon_m 0} \to P_{\varepsilon_m 0} \to P_0}{\neg P_{\epsilon_k} \to P_{\epsilon_m 0} \to P_0}}{\neg P_{\epsilon_k} \to (P_{\epsilon_m \epsilon_n} \to P_{\epsilon_n})}}{P_{\epsilon_m \epsilon_n} \to P_{\epsilon_n}}$$

Critical formulas:

$$\begin{aligned} P_{\epsilon_m 0} \to P_{\epsilon_m 0} & \quad \checkmark \\ (P_{\epsilon_m \epsilon_m 0} \to P_{\epsilon_m 0}) \to (P_{\epsilon_m \epsilon_m 0} \to P_{\epsilon_m 0}) & \quad \checkmark \\ (P_{\epsilon_m 0} \to P_0) \to (P_{\epsilon_m \epsilon_m 0} \to P_{\epsilon_m 0}) & \quad \checkmark \end{aligned}$$

ϵ-elimination:

- Try $\epsilon_k = \epsilon_n = 0$. Works unless $P_{\epsilon_m 0} \wedge \neg P_0$.
- But now we have a witness for $\exists k P_k$, so set $\epsilon_k = \epsilon_m = \epsilon_m 0$.

Finitary drinker's paradox I: For an arbitrary ϵ-term $\epsilon_m(\cdot)$ there exists some ϵ_n satisfying

$$P_{\epsilon_m \epsilon_n} \to P_{\epsilon_n}.$$

This can be computed by the algorithm

1. `Set` $\epsilon_n := 0$.
2. `Check` $P_{\epsilon_m 0} \to P_0$. `If true, END.`
3. `Else` $\epsilon_n := \epsilon_m 0$.

Finitary drinker's paradox I: For an arbitrary ϵ-term $\epsilon_m(\cdot)$ there exists some ϵ_n satisfying

$$P_{\epsilon_m \epsilon_n} \to P_{\epsilon_n}.$$

This can be computed by the algorithm

1. `Set` $\epsilon_n := 0$.
2. `Check` $P_{\epsilon_m 0} \to P_0$. `If true, END.`
3. `Else` $\epsilon_n := \epsilon_m 0$.

The term $\epsilon_m(\cdot)$ represents the 'proof theoretic environment', a measure of how we might *use* the drinkers paradox as a lemma, or more specifically, exactly when we need the $\forall$-axiom

$$\exists n \forall m (P_m \to P_n) \to \exists n (P_t \to P_n).$$

A 'modern' method: Gödel's functional (Dialectica) interpretation

Idea: A two stage translation: *Negative translation + Dialectica interpretation.*

A 'modern' method: Gödel's functional (Dialectica) interpretation

Idea: A two stage translation: *Negative translation + Dialectica interpretation.*

1. Eliminate classical reasoning by applying negative translation (can be more flexible here e.g. ignore atomic formulas).

2. Extract realizing terms for *Dialectica* interpretation of this formula. More complex than realizability - need to fully Skolemize implication:

$$
\begin{aligned}
(A \to B) &\rightsquigarrow (\exists x \forall y A_D(x,y) \to \exists u \forall v B_D(u,v)) \\
&\rightsquigarrow \forall x \exists u \forall v \exists y (A_D(x,y) \to B_D(u,v)) \\
&\rightsquigarrow \exists U, Y \forall x, v (A_D(x, Yxv) \to B_D(Ux, v))
\end{aligned}
$$

A 'modern' method: Gödel's functional (Dialectica) interpretation

Idea: A two stage translation: *Negative translation + Dialectica interpretation.*

1. Eliminate classical reasoning by applying negative translation (can be more flexible here e.g. ignore atomic formulas).

2. Extract realizing terms for *Dialectica* interpretation of this formula. More complex than realizability - need to fully Skolemize implication:

$$\begin{aligned}(A \to B) &\rightsquigarrow (\exists x \forall y A_D(x,y) \to \exists u \forall v B_D(u,v)) \\ &\rightsquigarrow \forall x \exists u \forall v \exists y (A_D(x,y) \to B_D(u,v)) \\ &\rightsquigarrow \exists U, Y \forall x, v (A_D(x, Yxv) \to B_D(Ux, v))\end{aligned}$$

Contraction problem: Interpretation of classical reasoning requires us to test atomic formulas and use case distinctions.

$$\dfrac{\exists k P_k \vee \forall k \neg P_k \qquad \begin{matrix}[\exists k P_k] \\ \vdots \\ \exists n \forall m (P_m \to P_n)\end{matrix} \qquad \begin{matrix}[\forall k \neg P_k] \\ \vdots \\ \exists n \forall m (P_m \to P_n)\end{matrix}}{\exists n \forall m (P_m \to P_n)}$$

$$
\dfrac{\exists k P_k \lor \forall k \neg P_k \qquad \begin{array}{c}[\exists k P_k] \\ \vdots \\ \neg\neg\exists n \forall m (P_m \to P_n)\end{array} \qquad \begin{array}{c}[\forall k \neg P_k] \\ \vdots \\ \neg\neg\exists n \forall m (P_m \to P_n)\end{array}}{\neg\neg\exists n \forall m (P_m \to P_n)}
$$

$$\dfrac{\exists k P_k \vee \forall k \neg P_k \qquad \begin{matrix} [P_k] \\ \vdots \\ P_{gk} \to P_k \end{matrix} \qquad \begin{matrix} [\forall k \neg P_k] \\ \vdots \\ \neg\neg\exists n \forall m (P_m \to P_n) \end{matrix}}{\neg\neg\exists n \forall m (P_m \to P_n)}$$

First branch:

$$\begin{aligned} &\exists k P_k \to \neg\neg\exists n \forall m (P_m \to P_n) \\ \rightsquigarrow &\exists k P_k \to \forall g^{\mathbb{N}\to\mathbb{N}} \exists n (P_{gn} \to P_n) \\ \rightsquigarrow &\forall g, k \exists n (P_k \to P_{gn} \to P_n) \\ \rightsquigarrow &\forall g, k (P_k \to P_{gk} \to P_k) \end{aligned}$$

$$\dfrac{\exists k P_k \vee \forall k \neg P_k \qquad \begin{array}{c}[P_k] \\ \vdots \\ P_{gk} \to P_k\end{array} \qquad \begin{array}{c}[\neg P_{g0}] \\ \vdots \\ P_{g0} \to P_0\end{array}}{\neg\neg\exists n \forall m (P_m \to P_n)}$$

First branch:

$$\begin{aligned}
&\exists k P_k \to \neg\neg\exists n \forall m (P_m \to P_n) \\
\rightsquigarrow &\exists k P_k \to \forall g^{\mathbb{N}\to\mathbb{N}} \exists n (P_{gn} \to P_n) \\
\rightsquigarrow &\forall g, k \exists n (P_k \to P_{gn} \to P_n) \\
\rightsquigarrow &\forall g, k (P_k \to P_{gk} \to P_k)
\end{aligned}$$

Second branch:

$$\begin{aligned}
&\forall k \neg P_k \to \neg\neg\exists n \forall m (P_m \to P_n) \\
\rightsquigarrow &\forall k \neg P_k \to \forall g \exists n (P_{gn} \to P_n) \\
\rightsquigarrow &\forall g \exists k, n (\neg P_k \to P_{gn} \to P_n) \\
\rightsquigarrow &\forall g (\neg P_{g0} \to P_{g0} \to P_0)
\end{aligned}$$

$$\dfrac{P_{g0} \vee \neg P_{g0} \qquad \begin{matrix}[P_{g0}] \\ \vdots \\ P_{g(g0)} \to P_{g0}\end{matrix} \qquad \begin{matrix}[\neg P_{g0}] \\ \vdots \\ P_{g0} \to P_0\end{matrix}}{\neg\neg\exists n \forall m (P_m \to P_n)}$$

First branch:

$$\begin{aligned}
&\exists k P_k \to \neg\neg\exists n \forall m (P_m \to P_n) \\
\rightsquigarrow &\exists k P_k \to \forall g^{\mathbb{N}\to\mathbb{N}} \exists n (P_{gn} \to P_n) \\
\rightsquigarrow &\forall g, k \exists n (P_k \to P_{gn} \to P_n) \\
\rightsquigarrow &\forall g, k (P_k \to P_{gk} \to P_k)
\end{aligned}$$

Second branch:

$$\begin{aligned}
&\forall k \neg P_k \to \neg\neg\exists n \forall m (P_m \to P_n) \\
\rightsquigarrow &\forall k \neg P_k \to \forall g \exists n (P_{gn} \to P_n) \\
\rightsquigarrow &\forall g \exists k, n (\neg P_k \to P_{gn} \to P_n) \\
\rightsquigarrow &\forall g (\neg P_{g0} \to P_{g0} \to P_0)
\end{aligned}$$

$$
\cfrac{P_{g0} \vee \neg P_{g0} \qquad \begin{array}{c}[P_{g0}] \\ \vdots \\ P_{g(g0)} \to P_{g0}\end{array} \qquad \begin{array}{c}[\neg P_{g0}] \\ \vdots \\ P_{g0} \to P_0\end{array}}{P_{g(Ng)} \to P_{Ng}}
$$

Solved by

$$
Ng := \begin{cases} 0 & \text{if } \neg P_{g0} \\ g0 & \text{if } P_{g0} \end{cases}
$$

Finitary drinker's paradox II: For an arbitrary function $g \colon \mathbb{N} \to \mathbb{N}$ there exists some $N \colon (\mathbb{N} \to \mathbb{N}) \to \mathbb{N}$ satisfying

$$P_{g(Ng)} \to P_{Ng}.$$

This can be defined as

$$Ng := \begin{cases} 0 & \text{if } \neg P_{g0} \\ g0 & \text{otherwise.} \end{cases}$$

Finitary drinker's paradox II: For an arbitrary function $g\colon \mathbb{N} \to \mathbb{N}$ there exists some $N\colon (\mathbb{N} \to \mathbb{N}) \to \mathbb{N}$ satisfying

$$P_{g(Ng)} \to P_{Ng}.$$

This can be defined as

$$Ng := \begin{cases} 0 & \text{if } \neg P_{g0} \\ g0 & \text{otherwise.} \end{cases}$$

The proof theoretic environment is represented by an explicit 'counterexample function' g. Any instance of the drinkers paradox in a bigger proof will involve a concrete instantiation gv of g:

$$\begin{gathered} \exists n \forall m (P_m \to P_n) \to \exists u \forall v B(u, v) \\ \rightsquigarrow \exists U, g \forall n, v ((P_{gvn} \to P_n) \to B(Un, v)) \end{gathered}$$

and hence $\forall v B(U(N(gv)), v)$ holds.

General finitary drinker's paradox: There exists an approximate witness $\mathcal{N}$ to $\exists n \forall m (P_n \to P_m)$, that works relative to any environment $\mathcal{M}$ (representing $\forall m$).

Technique	$\mathcal{N}$	$\mathcal{M}$
ϵ-calculus	$\left[\begin{array}{l} \epsilon_n := 0 \\ \texttt{Check } P_{\epsilon_m 0} \to P_0\texttt{. If true, END.} \\ \texttt{Else } \epsilon_n := \epsilon_m 0 \end{array}\right.$	$\epsilon_m(\cdot)$
Dialectica	$Ng := \begin{cases} 0 & \text{if } \neg P_{g0} \\ g0 & \text{otherwise} \end{cases}$	$g \colon \mathbb{N} \to \mathbb{N}$

Both methods carry out 'learning', but in completely different frameworks:

What is the general idea I'm getting at?

What is the general idea I'm getting at?

One can interpret non-constructive theorems by 'finitized' versions of those theorems:

A : there exists an ideal object x.

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

WHAT IS THE GENERAL IDEA I'M GETTING AT?

One can interpret non-constructive theorems by 'finitized' versions of those 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.

What is the general idea I'm getting at?

One can interpret non-constructive theorems by 'finitized' versions of those 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.

Technique	approxmation	algorithm
ϵ-calculus	ϵ-terms	ϵ-elimination
Dialectica	negative translation + Skolemisation	**THIS TALK**

Why should we care about this?

The Dialectica interpretation has proven itself to be one of the most powerful methods for extracting programs from proofs. In particular, it is the basic technique that underlies the whole *proof mining* program.

Why should we care about this?

The Dialectica interpretation has proven itself to be one of the most powerful methods for extracting programs from proofs. In particular, it is the basic technique that underlies the whole *proof mining* program.

However, for any but the simplest proofs, extracted programs are hugely complex, and their behaviour can be extremely difficult to understand. We want to improve this situation.

Why should we care about this?

The Dialectica interpretation has proven itself to be one of the most powerful methods for extracting programs from proofs. In particular, it is the basic technique that underlies the whole *proof mining* program.

However, for any but the simplest proofs, extracted programs are hugely complex, and their behaviour can be extremely difficult to understand. We want to improve this situation.

In this talk, I hope to show that the concept of learning underlies the Dialectica interpretation, and that studying this can lead to:

1. Refined interpretations which extract more intuitive programs from proofs *(learning)*;
2. Interesting problems in computability theory *(limits)*;
3. Potentially new applications in computer science *(loops)*.

A (highly selective) list of works on the connection between classical logic and learning that are particularly relevant here:

- **Hilbert (1930s).** Epsilon calculus and elimination procedure.
- **Coquand et al. (1990s).** Novikoff's calculus, and truth as a 'winning strategy' in a game between quantifiers. In particular, a winning strategy for countable choice (1998).
- **Avigad (2002).** Formalisation of learning implicit in epsilon calculus via 'update procedures'.
- **Aschieri, Berardi et al. (c. 2005-).** Development of explicit 'learning-based' computational interpretations of classical logic.
- **Kohlenbach, Safarik (2013).** A quantitative analysis of learning procedures extracted from convergence proofs.

Learning!

How does the Dialectica interpretation interpret a Π_3-theorem?

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

How does the Dialectica interpretation interpret 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:

How does the Dialectica interpretation interpret 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:

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

How does the Dialectica interpretation interpret 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:

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

How does the Dialectica interpretation interpret 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:

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

Over classical logic

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

but we can realize the R.H.S.

INFINITARY. $\exists n \in \mathbb{N} \forall m \in \mathbb{N}(P_m \to P_n)$.

INFINITARY. $\exists n \in \mathbb{N} \forall m \in \mathbb{N}(P_m \to P_n)$.

FINITARY. $\forall g : \mathbb{N} \to \mathbb{N} \exists n (P_{gn} \to P_n)$

INFINITARY. $\exists n \in \mathbb{N} \forall m \in \mathbb{N}(P_m \to P_n)$.

FINITARY. $\forall g : \mathbb{N} \to \mathbb{N} \exists n(P_{gn} \to P_n)$

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

$$n := \begin{cases} 0 & \text{if } \neg P_{g0} \\ g0 & \text{otherwise.} \end{cases}$$

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

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

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 $g\colon \mathbb{N} \to \mathbb{N}$ there exists some $y \geq x$ such that

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

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 $g\colon \mathbb{N} \to \mathbb{N}$ there exists some $y \geq x$ such that

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

We can compute y by learning, as follows:

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

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

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 $g \colon \mathbb{N} \to \mathbb{N}$ there exists Y such that for any finite increasing sequence

$$0 \leq a_0 \leq \ldots \leq a_Y \leq 1$$

there exists some y with $0 \leq y < y + g(y) \leq Y$ such that

$$|a_i - a_j| < 2^{-x}$$

for all $y \leq i, j \leq g(y)$.

We can show that $Y = (\lambda i.i + g(i))^{(2^x)}(0)$.

The intuition behind learning algorithms

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

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 $g(x_0)$.

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 $g(x_0)$.

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

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 $g(x_0)$.

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

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

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 $\mathcal{L} = (\mathsf{G}, g, \oplus)$ where

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

Learning algorithms - a formal definition

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

- $\mathsf{G} : X \to \mathbb{B}$ is a decidable predicate which tests whether an element $x \in X$ is 'good';
- $g : X \to L$ 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 $x \oplus g(x)$;

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

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

Learning algorithms - a formal definition

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

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

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

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

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).

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

$$\forall g \exists x A'_g(x, g)$$

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

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

for some x_0.

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

$$\forall g \exists x A'_g(x, g)$$

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

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

for some x_0.

This idea is made more precise in (P. 2016), 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 this talk I will just give some illustrations.

EXAMPLE 1: The quantifier-free minimum principle

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

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

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

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

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

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

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

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

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

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

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

Then the ND interpretation of QFMin, given by

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

is realized by

$$\lambda x, g\ .\ \lim \mathcal{L}_g[x].$$

REMARK. A general result dealing with well-founded induction for arbitrary formulas and relations $\prec$ is given in (P. 2016), inspired by (Schwichtenberg 2008).

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

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

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

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

This is just the Euclidean algorithm!

Limits, or alternatively: are there any interesting examples of learning procedures?

Let's go back (one final time!) to the drinkers paradox:

$$(\exists n \in \mathbb{N})(\forall m \in \mathbb{N})(P_m \to P_n).$$

Let's go back (one final time!) to the drinkers paradox:

$$(\exists n \in \mathbb{N})(\forall m \in \mathbb{N})(P_m \to P_n).$$

which we know is interpreted as

$$\forall g \exists n (P_{gn} \to P_n)$$

i.e. "for any g there is a g-approximation to the ideal object n which works for gn".

Let's go back (one final time!) to the drinkers paradox:

$$(\exists n \in \mathbb{N})(\forall m \in \mathbb{N})(P_m \to P_n).$$

which we know is interpreted as

$$\forall g \exists n (P_{gn} \to P_n)$$

i.e. "for any g there is a g-approximation to the ideal object n which works for gn".

It is realized by the two-step learning procedure

$$n := \begin{cases} 0 & \text{if } \neg P_{g0} \\ g0 & \text{if } P_{g0} \end{cases}$$

i.e. either our default guess 0 is true, or it is false, and from its falsity we are able to collect a piece of constructive information about P, namely that P_{g0} is true.

But now suppose that we have an infinite sequence of predicates $P(k)$. Then by classical logic we have

$$(\forall k)(\exists n)(\forall m)(P(k)_m \to P(k)_n)$$

and by **countable choice** we obtain

$$(\exists f \in \mathbb{N} \to \mathbb{N})(\forall m, k)(P(k)_m \to P(k)_{f(k)}).$$

But now suppose that we have an infinite sequence of predicates $P(k)$. Then by classical logic we have

$$(\forall k)(\exists n)(\forall m)(P(k)_m \to P(k)_n)$$

and by **countable choice** we obtain

$$(\exists f \in \mathbb{N} \to \mathbb{N})(\forall m, k)(P(k)_m \to P(k)_{f(k)}).$$

To give this a computational interpretation, we double negate:

$$\neg\neg(\exists f)(\forall m, k)(P(k)_m \to P(k)_{f(k)}).$$

and then Skolemise:

$$(\forall \phi, \omega : \mathbb{N}^{\mathbb{N}} \to \mathbb{N})(\exists f)(P(\omega f)_{\phi f} \to P(\omega f)_{f(\omega f)}).$$

"For any ϕ, ω there is a (ϕ, ω)-approximation to f which works for ϕf at point ωf."

Goal: Given ϕ, ω produce f satisfying

$$P(\omega f)_{\phi f} \to P(\omega f)_{f(\omega f)}.$$

Goal: Given ϕ, ω produce f satisfying

$$P(\omega f)_{\phi f} \to P(\omega f)_{f(\omega f)}.$$

Let's try the following learning procedure, of 'unbounded' length:

$$\begin{aligned}
f_0 &= 0, 0, 0, 0, 0, \ldots \\
f_1 &= 0, 0, \underbrace{\phi f_0}_{\omega f_0}, 0, 0, \ldots \\
f_2 &= 0, 0, \underbrace{\phi f_0}_{\omega f_0}, 0, \underbrace{\phi f_1}_{\omega f_1}, \ldots \\
f_3 &= 0, \underbrace{\phi f_2}_{\omega f_2}, \underbrace{\phi f_0}_{\omega f_0}, 0, \underbrace{\phi f_1}_{\omega f_1}, \ldots \\
&\ldots
\end{aligned}$$

We terminate the procedure unless

$$\omega f_i \notin \mathrm{dom}(f_i) \wedge P(\omega f_i)_{\phi f_i},$$

where $\mathrm{dom}(f_i) = \{\omega f_0, \ldots, \omega f_{i-1}\}$.

Informal idea: For all $k \in \mathrm{dom}(f_i)$ we have $P(k)_{f_i(k)}$, and so we're progressively building a better approximation of f.

At each stage, either:

✓ $\omega f_i \in \mathrm{dom}(f_i)$, and we're done (satisfy ω-test);

✓ $\neg P(\omega f_i)_{\phi f_i}$, and we're done (satisfy ϕ-test);

X $\omega f_i \notin \mathrm{dom}(f_i) \wedge P(\omega f_i)_{\phi f_i}$ and we have learned a new piece of constructive information about P, so we update our approximation:

$$f_{i+1} = f_i[\omega f_i \mapsto \phi f_i].$$

Informal idea: For all $k \in \mathrm{dom}(f_i)$ we have $P(k)_{f_i(k)}$, and so we're progressively building a better approximation of f.

At each stage, either:

✓ $\omega f_i \in \mathrm{dom}(f_i)$, and we're done (satisfy ω-test);

✓ $\neg P(\omega f_i)_{\phi f_i}$, and we're done (satisfy ϕ-test);

X $\omega f_i \notin \mathrm{dom}(f_i) \wedge P(\omega f_i)_{\phi f_i}$ and we have learned a new piece of constructive information about P, so we update our approximation:

$$f_{i+1} = f_i[\omega f_i \mapsto \phi f_i].$$

Whenever ω is a continuous functional, it only looks at a finite part of its input, so we will eventually reach some N such that

$$\omega f_N \in \mathrm{dom}(f_N) = \{\omega f_0, \ldots, \omega f_{N-1}\}.$$

Therefore the learning algorithm terminates.

$$\forall k \neg\neg\exists n \forall m (P(k)_m \to P(k)_n)$$

$$\Downarrow$$

$$n := \begin{cases} 0 & \text{if } \neg P(k)_{g0} \\ g0 & \text{otherwise} \end{cases}$$

$$\forall k \neg\neg\exists n \forall m (P(k)_m \to P(k)_n)$$

$$\Downarrow$$

$$n := \begin{cases} 0 & \text{if } \neg P(k)_{g0} \\ g0 & \text{otherwise} \end{cases}$$

$$\neg\neg(\exists f \in \mathbb{N} \to \mathbb{N})(\forall m, k)(P(k)_m \to P(k)_{f(k)})$$

$$\Downarrow$$

$$f := \begin{cases} \underbrace{\mathbf{0}}_{f_0} = \lambda i.0 & \text{if } \omega f_0 \in \mathrm{dom}(f_0) \vee \neg P(\omega f_0)_{\phi f_0} \\ \underbrace{f_0[\omega f_0 \mapsto \phi f_0]}_{f_1} & \text{if } \omega f_1 \in \mathrm{dom}(f_1) \vee \neg P(\omega f_1)_{\phi f_1} \\ \underbrace{f_1[\omega f_1 \mapsto \phi f_1]}_{f_2} & \text{if } \omega f_2 \in \mathrm{dom}(f_2) \vee \neg P(\omega f_2)_{\phi f_2} \\ \dots & \dots \end{cases}$$

This is an instance of the double negation shift

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

which has a (partial) functional interpretation given by

$$\forall n, g\exists x A_n(x,g(x))\to\forall\omega,\phi\exists\alpha^{X^{\mathbb{N}}}A_{\omega\alpha}(\alpha(\omega\alpha),\phi\alpha)$$

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

This is an instance of the double negation shift

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

which has a (partial) functional interpretation given by

$$\underbrace{\forall n, g \exists x A_n(x, g(x))}_{(\mathcal{L}_{n,g})_{n<\infty}} \to \underbrace{\forall \omega, \phi \exists \alpha^{X^{\mathbb{N}}} A_{\omega\alpha}(\alpha(\omega\alpha), \phi\alpha)}_{\mathcal{L}_{\infty,(\omega,\phi)}}$$

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

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

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

This is an instance of the double negation shift

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

which has a (partial) functional interpretation given by

$$\underbrace{\forall n, g \exists x A_n(x, g(x))}_{(\mathcal{L}_{n,g})_{n<\infty}} \to \underbrace{\forall \omega, \phi \exists \alpha^{X^{\mathbb{N}}} A_{\omega\alpha}(\alpha(\omega\alpha), \phi\alpha)}_{\mathcal{L}_{\infty,(\omega,\phi)}}$$

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

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

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

YES! Details in (P. 2016)... I gave just a very simple illustration here!

Remark. The realizers we obtain for the functional interpretation of comprehension principles using learning procedures is different from those we obtain using bar recursion. In this context, bar recursion is equivalent to a form of 'forgetful' learning, which erases information it has learned above the point being updated e.g.

$$
\begin{aligned}
f_0 &= 0, 0, 0, 0, 0 \ldots \\
f_1 &= 0, 0, \underbrace{\phi f_0}_{\omega f_0}, 0, \ldots \\
f_2 &= 0, 0, \phi f_0, 0, \underbrace{\phi f_1}_{\omega f_1}, \ldots \\
f_3 &= 0, \underbrace{\phi f_2}_{\omega f_2}, 0, 0, 0, \ldots \\
&\ldots
\end{aligned}
$$

Further detals in (P. 2016).

Historical remark. Gödel developed his functional interpretation over a period of 30 years, finally publishing it in 1958. The original paper dealt only with Peano arithmetic, and was extended to analysis by Spector in 1962.

Spector showed that the functional interpretation of the double negation shift (and hence the ND interpretation of countable choice) could be realized using bar recursion in all finite types.

However, Spector's paper was left unfinished - in particular, Section 12.1 sketches an *alternative* to bar recursion for a very simple case of DNS, namely a learning algorithm similar to that presented here. Kreisel received this as a letter, writing:

"The typescript ... consists of sections up to and including 12.1 of the present paper, with about half a page (crossed out) of a projected 12.2. This last half page states that the proof of the Gödel translation of axiom F [the DNS] would use a generalization of Hilbert's substitution method as illustrated in the special case of 12.1. However Spector's notes do not contain any details, so that it is not quite clear how to reconstruct the proof he had in mind."

You may be wondering... What does any of this have to do with Zorn's lemma? (cf. my Abstract!)

You may be wondering... What does any of this have to do with Zorn's lemma? (cf. my Abstract!)

The two main kinds of non-constructive proof we have looked at are (a) minimum principles, (b) countable choice.

You may be wondering... What does any of this have to do with Zorn's lemma? (cf. my Abstract!)

The two main kinds of non-constructive proof we have looked at are (a) minimum principles, (b) countable choice.

But countable choice is provable using a weak form of Zorn's lemma, which is in turn just a 'minimum principle' over chain-complete partial orders. More precisely, a choice function is a minimum element of the set of partial choice functions ordered by $f \sqsupset g$ whenever f is an extension of g.

You may be wondering... What does any of this have to do with Zorn's lemma? (cf. my Abstract!)

The two main kinds of non-constructive proof we have looked at are (a) minimum principles, (b) countable choice.

But countable choice is provable using a weak form of Zorn's lemma, which is in turn just a 'minimum principle' over chain-complete partial orders. More precisely, a choice function is a minimum element of the set of partial choice functions ordered by $f \sqsupset g$ whenever f is an extension of g.

So our approach allows us to unify the computational interpretations of arithmetic (induction $\sim$ minimum principle) and analysis (comprehension $\sim$ Zorn's lemma).

You may be wondering... What does any of this have to do with Zorn's lemma? (cf. my Abstract!)

The two main kinds of non-constructive proof we have looked at are (a) minimum principles, (b) countable choice.

But countable choice is provable using a weak form of Zorn's lemma, which is in turn just a 'minimum principle' over chain-complete partial orders. More precisely, a choice function is a minimum element of the set of partial choice functions ordered by $f \sqsupset g$ whenever f is an extension of g.

So our approach allows us to unify the computational interpretations of arithmetic (induction $\sim$ minimum principle) and analysis (comprehension $\sim$ Zorn's lemma).

Open question. Can we give a computational interpretation to stronger forms of Zorn's lemma (e.g. minimal bad sequence arguments)? But this is a different talk...

And to conclude... Loops!

Learning procedures are useful because they give extracted programs a more 'imperative' feel, allowing us to understand how these programs compute realizers.

In fact, a learning procedure is nothing more than a while loop - the following imperative program computes $\lim \mathcal{L}_g[x_0]$:

```
y := x_0
while ¬G(y)
   y := y ⊕ g(y)
return y
```

Learning procedures are useful because they give extracted programs a more 'imperative' feel, allowing us to understand how these programs compute realizers.

In fact, a learning procedure is nothing more than a while loop - the following imperative program computes $\lim \mathcal{L}_g[x_0]$:

$$\left[\begin{array}{l} \mathtt{y} := \mathtt{x}_0 \\ \mathtt{while}\ \neg \mathsf{G}(\mathtt{y}) \\ \quad \mathtt{y} := \mathtt{y} \oplus \mathtt{g}(\mathtt{y}) \\ \mathtt{return}\ \mathtt{y} \end{array}\right.$$

Extracted programs can be incredible simple. To compute a realizer for the functional interpretation of our instance of comprehension, this suffices:

$$\left[\begin{array}{l} \mathtt{f} := [] \\ \mathtt{while}\ \omega\mathtt{f} \notin \mathrm{dom}(\mathtt{f}) \wedge \mathtt{P}(\omega\mathtt{f})_{\phi\mathtt{f}} \\ \quad \mathtt{f} := \mathtt{f}[\omega\mathtt{f} \mapsto \phi\mathtt{f}] \\ \mathtt{return}\ \mathtt{f} \end{array}\right.$$

Compare this to bar recursion!

QUESTION. Can we develop an **imperative** functional interpretation?

QUESTION. Can we develop an **imperative** functional interpretation?

Usually we have

$$\begin{cases} \text{Predicate logic} + \text{Induction} \vdash A \\ \Rightarrow \text{can extract a term } t \text{ of system } T \text{ s.t. } T \vdash A_D(t, x) \end{cases}$$

QUESTION. Can we develop an **imperative** functional interpretation?

Usually we have

$$\begin{cases} \text{Predicate logic} + \text{Induction} \vdash A \\ \Rightarrow \text{can extract a term } t \text{ of system } T \text{ s.t. } T \vdash A_D(t,x) \end{cases}$$

An imperative version of the interpretation might look something like the following:

$$\begin{cases} \text{Predicate logic} + \text{Hoare-style rules} \vdash A \Rightarrow \\ \text{can extract a program } S \text{ s.t. Hoare logic } \vdash \{I\}\ S\ \{A_D(y,x)\} \end{cases}$$

Obviously there are many subtleties here , notably how to incorporate higher-order features (which will always be essential).

But the benefits and new applications could be worth the effort!

Adapting the functonal interpretation in an imperative setting would:

Adapting the functonal interpretation in an imperative setting would:

1. Take further step towards making the connecting between classical logic and learning more precise, continuing the line of thought begun by Hilbert and further developed by Coquand, Aschieri et al. etc.

Adapting the functonal interpretation in an imperative setting would:

1. Take further step towards making the connecting between classical logic and learning more precise, continuing the line of thought begun by Hilbert and further developed by Coquand, Aschieri et al. etc.

2. Enable us to extract more efficient, readable and intuitive programs from proofs in mathematics.

Adapting the functonal interpretation in an imperative setting would:

1. Take further step towards making the connecting between classical logic and learning more precise, continuing the line of thought begun by Hilbert and further developed by Coquand, Aschieri et al. etc.

2. Enable us to extract more efficient, readable and intuitive programs from proofs in mathematics.

3. Potentially lead to new applications in computer science by providing us with a technique for formally extracting verified imperative programs.

Adapting the functonal interpretation in an imperative setting would:

1. Take further step towards making the connecting between classical logic and learning more precise, continuing the line of thought begun by Hilbert and further developed by Coquand, Aschieri et al. etc.

2. Enable us to extract more efficient, readable and intuitive programs from proofs in mathematics.

3. Potentially lead to new applications in computer science by providing us with a technique for formally extracting verified imperative programs.

I'm particularly interested in the last point - this has been explored by (Berger et al. 2014), but there is a huge scope for further work, which could lead to applications in the real world...