Proof interpretations: a modern perspective

Lecture 5 - Proof interpretations today

Anupam Das & Thomas Powell

University of Copenhagen & Technische Universität Darmstadt

North American Summer School on Logic, Language, and Information

Carnegie Mellon University

29 June 2018

These slides are available at `http://www.anupamdas.com/nasslli18`.

Outline

The potential of program extraction

We have already seen some examples of witness extraction from $\forall\exists$ statements, our running example being

Theorem

There exists a function $X : \mathbb{N} \to \mathbb{N}$ such that for all n we have $X(n) \geq n$ and $X(n)$ prime.

But you don't need sophisticated proof theoretic techniques to be able to do this. So are there examples where the formal analysis of a proof can yield genuinely new numerical information from proofs?

The potential of program extraction

We have already seen some examples of witness extraction from $\forall\exists$ statements, our running example being

Theorem

There exists a function $X : \mathbb{N} \to \mathbb{N}$ such that for all n we have $X(n) \geq n$ and $X(n)$ prime.

But you don't need sophisticated proof theoretic techniques to be able to do this. So are there examples where the formal analysis of a proof can yield genuinely new numerical information from proofs?

The answer is an emphatic YES. This is the so-called 'proof mining' program.

Central to the success of proof mining program are the following phenomena:

- One can typically extract a witnesses for $\forall\exists$ statements even when the underlying proofs are highly non-constructive;
- Certain mathematical principles, particularly forms of *compactness*, do not contribute to the complexity of extracted bounds, leading to surprisingly simple polynomial bounds from proofs which employ heavy machinery from analysis.

A brief history of proof mining

A brief history of proof mining

- Pioneered by Kreisel in the 1950s, who proposed 'unwinding' constructive content from proofs using proof theoretic methods. Case studies in number theory and abstract algebra.

A brief history of proof mining

- Pioneered by Kreisel in the 1950s, who proposed 'unwinding' constructive content from proofs using proof theoretic methods. Case studies in number theory and abstract algebra.
- In the 1980s, both Girard and Luckhardt carry out case studies and obtain bounds (van der Waerden's theorem and Roth's theorem respectively)

A brief history of proof mining

- Pioneered by Kreisel in the 1950s, who proposed 'unwinding' constructive content from proofs using proof theoretic methods. Case studies in number theory and abstract algebra.
- In the 1980s, both Girard and Luckhardt carry out case studies and obtain bounds (van der Waerden's theorem and Roth's theorem respectively)
- From 1990s onwards, Kohlenbach finds numerous applications, in approximation theory and fixed point theory in particular. Proof mining takes off!

A brief history of proof mining

- Pioneered by Kreisel in the 1950s, who proposed 'unwinding' constructive content from proofs using proof theoretic methods. Case studies in number theory and abstract algebra.
- In the 1980s, both Girard and Luckhardt carry out case studies and obtain bounds (van der Waerden's theorem and Roth's theorem respectively)
- From 1990s onwards, Kohlenbach finds numerous applications, in approximation theory and fixed point theory in particular. Proof mining takes off!
- In the 2010s Avigad, Towsner and others analyse convergence proofs in ergodic theory.

A brief history of proof mining

- Pioneered by Kreisel in the 1950s, who proposed 'unwinding' constructive content from proofs using proof theoretic methods. Case studies in number theory and abstract algebra.
- In the 1980s, both Girard and Luckhardt carry out case studies and obtain bounds (van der Waerden's theorem and Roth's theorem respectively)
- From 1990s onwards, Kohlenbach finds numerous applications, in approximation theory and fixed point theory in particular. Proof mining takes off!
- In the 2010s Avigad, Towsner and others analyse convergence proofs in ergodic theory.
- In the last few years, Kohlenbach and his students find applications in convex optimization.

A brief history of proof mining

- Pioneered by Kreisel in the 1950s, who proposed 'unwinding' constructive content from proofs using proof theoretic methods. Case studies in number theory and abstract algebra.
- In the 1980s, both Girard and Luckhardt carry out case studies and obtain bounds (van der Waerden's theorem and Roth's theorem respectively)
- From 1990s onwards, Kohlenbach finds numerous applications, in approximation theory and fixed point theory in particular. Proof mining takes off!
- In the 2010s Avigad, Towsner and others analyse convergence proofs in ergodic theory.
- In the last few years, Kohlenbach and his students find applications in convex optimization.
- **2018: Where to next?**

Example: Uniqueness of best approximation

Theorem

Let $n \in \mathbb{N}$ and $f \in C[0,1]$ be fixed. Let

$$dist(f, P_n) := \inf_{p \in P_n} \|f - p\|$$

where P_n is the space of all polynomials with degree $\leq n$. Then there exists a polynomial of best approximation i.e. a polynomial p^* such that

$$\|f - p^*\| = dist(f, P_n),$$

and moreover, this polynomial is unique i.e. for all $p_1, p_2 \in P_n$

$$\bigwedge_{i=1,2} (\|f - p_i\| = dist(f, P_n)) \to p_1 = p_2.$$

A proof theoretic analysis of uniqueness

Let's look a bit more closely at uniqueness:

$$\forall n \in \mathbb{N} \forall f \in C[0,1] \forall p_1, p_2 \in P_n \left(\bigwedge_{i=1,2} (\|f - p_i\| = \mathrm{dist}(f, P_n)) \to p_1 = p_2 \right).$$

A proof theoretic analysis of uniqueness

Let's look a bit more closely at uniqueness:

$$\forall n \in \mathbb{N} \forall f \in C[0,1] \forall p_1, p_2 \in P_n \left(\bigwedge_{i=1,2} (\|f - p_i\| = \operatorname{dist}(f, P_n)) \to p_1 = p_2 \right).$$

Now, equality $=$ over the real numbers is actually a $\forall$-statement and so written out fully, uniqueness becomes

$$\begin{cases} \forall n \in \mathbb{N} \forall f \in C[0,1] \forall p_1, p_2 \in P_n \\ \left(\forall j \bigwedge_{i=1,2} (\|f - p_i\| - \operatorname{dist}(f, P_n) < 2^{-j}) \to \forall k \, \|p_1 - p_2\| < 2^{-k} \right). \end{cases}$$

A proof theoretic analysis of uniqueness

Let's look a bit more closely at uniqueness:

$$\forall n \in \mathbb{N} \forall f \in C[0,1] \forall p_1, p_2 \in P_n \left(\bigwedge_{i=1,2} (\|f - p_i\| = \operatorname{dist}(f, P_n)) \to p_1 = p_2 \right).$$

Now, equality $=$ over the real numbers is actually a $\forall$-statement and so written out fully, uniqueness becomes

$$\begin{cases} \forall n \in \mathbb{N} \forall f \in C[0,1] \forall p_1, p_2 \in P_n \\ \left(\forall j \bigwedge_{i=1,2} (\|f - p_i\| - \operatorname{dist}(f, P_n) < 2^{-j}) \to \forall k \, \|p_1 - p_2\| < 2^{-k} \right). \end{cases}$$

The (partial) functional interpretation of this is the following:

$$\begin{cases} \forall n, k \in \mathbb{N} \forall f \in C[0,1] \forall p_1, p_2 \in P_n \exists j \\ \left(\bigwedge_{i=1,2} (\|f - p_i\| - \operatorname{dist}(f, P_n) < 2^{-j}) \to \|p_1 - p_2\| < 2^{-k} \right). \end{cases}$$

A modulus of uniqueness

In the case of both the uniform norm and the L_1 norm, it is possible to extract a term Φ of System T such that

$$\begin{cases} \forall n, k \in \mathbb{N} \forall f \in C[0,1] \forall p_1, p_2 \in P_n \exists j \\ \left(\bigwedge_{i=1,2} (\|f - p_i\| - \operatorname{dist}(f, P_n) < 2^{-\Phi(f,n,k)}) \to \|p_1 - p_2\| < 2^{-k} \right). \end{cases}$$

where Φ is independent of p_1, p_2.

Remark. Φ is known as the modulus of uniqueness.

A modulus of uniqueness

In the case of both the uniform norm and the L_1 norm, it is possible to extract a term Φ of System T such that

$$\begin{cases} \forall n, k \in \mathbb{N} \forall f \in C[0,1] \forall p_1, p_2 \in P_n \exists j \\ \left(\bigwedge_{i=1,2} (\|f - p_i\| - \operatorname{dist}(f, P_n) < 2^{-\Phi(f,n,k)}) \to \|p_1 - p_2\| < 2^{-k} \right). \end{cases}$$

where Φ is independent of p_1, p_2.

Remark. Φ is known as the modulus of uniqueness.

Explicit moduli of uniqueness are given in the following papers:

- de La Vallée Poussin's proof of uniqueness of best Chebychev approximation [Kohlenbach, 1993a];
- Young's proof of uniqueness of best Chebychev approximation [Kohlenbach, 1993b];
- Cheney's proof of uniqueness of best L_1 approximation [Kohlenbach and Oliva, 2003a].

In some cases these results even improved known results in the literature.

More recent work

For a comprehensive account of proof mining see [Kohlenbach, 2008] (the standard text on the subject).

More recent work

For a comprehensive account of proof mining see [Kohlenbach, 2008] (the standard text on the subject).

Following early success in approximation theory, proof interpretations have been used to extract new quantitative information, and establish abstract generalizations, in the following areas in particular:

- Fixed point theory
- Ergodic theory
- Convex optimization

More recent work

For a comprehensive account of proof mining see [Kohlenbach, 2008] (the standard text on the subject).

Following early success in approximation theory, proof interpretations have been used to extract new quantitative information, and establish abstract generalizations, in the following areas in particular:

- Fixed point theory
- Ergodic theory
- Convex optimization

For individual expository articles see e.g.

More recent work

For a comprehensive account of proof mining see [Kohlenbach, 2008] (the standard text on the subject).

Following early success in approximation theory, proof interpretations have been used to extract new quantitative information, and establish abstract generalizations, in the following areas in particular:

- Fixed point theory
- Ergodic theory
- Convex optimization

For individual expository articles see e.g.

- Kohlenbach, U. and Oliva, P. (2003b). A systematic way of analyzing proofs in mathematics.
  Proceedings of the Steklov Institute of Mathematics, 242:136–164

More recent work

For a comprehensive account of proof mining see [Kohlenbach, 2008] (the standard text on the subject).

Following early success in approximation theory, proof interpretations have been used to extract new quantitative information, and establish abstract generalizations, in the following areas in particular:

- Fixed point theory
- Ergodic theory
- Convex optimization

For individual expository articles see e.g.

- Kohlenbach, U. and Oliva, P. (2003b). A systematic way of analyzing proofs in mathematics.
  Proceedings of the Steklov Institute of Mathematics, 242:136–164
- Avigad, J. (2009). The metamathematics of ergodic theory.
  Annals of Pure and Applied Logic, 157:64–76

More recent work

For a comprehensive account of proof mining see [Kohlenbach, 2008] (the standard text on the subject).

Following early success in approximation theory, proof interpretations have been used to extract new quantitative information, and establish abstract generalizations, in the following areas in particular:

- Fixed point theory
- Ergodic theory
- Convex optimization

For individual expository articles see e.g.

- Kohlenbach, U. and Oliva, P. (2003b). A systematic way of analyzing proofs in mathematics.
  Proceedings of the Steklov Institute of Mathematics, 242:136–164
- Avigad, J. (2009). The metamathematics of ergodic theory.
  Annals of Pure and Applied Logic, 157:64–76
- Kohlenbach, U. Proof theoretic methods in nonlinear analysis.
  To appear in: Proc. Int. Cong. of Math. - ICM 2018

How are proof theoretic tools applied to new areas?

Key steps:

How are proof theoretic tools applied to new areas?

Key steps:

1. Are there theorems in this area which have the right logical structure? What kind of information could I hope to extract?

How are proof theoretic tools applied to new areas?

Key steps:

1. Are there theorems in this area which have the right logical structure? What kind of information could I hope to extract?
2. How do I formalize the proofs? How do I represent the underlying spaces?

How are proof theoretic tools applied to new areas?

Key steps:

1. Are there theorems in this area which have the right logical structure? What kind of information could I hope to extract?
2. How do I formalize the proofs? How do I represent the underlying spaces?
3. Analyse some concrete proofs.

How are proof theoretic tools applied to new areas?

Key steps:

1. Are there theorems in this area which have the right logical structure? What kind of information could I hope to extract?
2. How do I formalize the proofs? How do I represent the underlying spaces?
3. Analyse some concrete proofs.
4. What is going on more generally? Can these proofs be expressed in an abstract logical framework?

How are proof theoretic tools applied to new areas?

Key steps:

1. Are there theorems in this area which have the right logical structure? What kind of information could I hope to extract?
2. How do I formalize the proofs? How do I represent the underlying spaces?
3. Analyse some concrete proofs.
4. What is going on more generally? Can these proofs be expressed in an abstract logical framework?
5. Develop new metatheorems which *guarantee* that, under certain conditions, programs can be extracted.

How are proof theoretic tools applied to new areas?

Key steps:

1. Are there theorems in this area which have the right logical structure? What kind of information could I hope to extract?
2. How do I formalize the proofs? How do I represent the underlying spaces?
3. Analyse some concrete proofs.
4. What is going on more generally? Can these proofs be expressed in an abstract logical framework?
5. Develop new metatheorems which *guarantee* that, under certain conditions, programs can be extracted.

Potential new areas:

- Number theory?
- Probability theory?
- Financial mathematics?

Outline

The monotone convergence theorem

Recall in the last lecture we discussed the *monotone convergence principle*:

Theorem

Let (x_n) be a nondecreasing sequence of rational numbers in $[0,1]$. Then

$$\forall k \exists n \forall m (|x_{n+m} - x_n| \leq 2^{-k}).$$

The monotone convergence theorem

Recall in the last lecture we discussed the *monotone convergence principle*:

Theorem

Let (x_n) be a nondecreasing sequence of rational numbers in $[0,1]$. Then

$$\forall k \exists n \forall m (|x_{n+m} - x_n| \leq 2^{-k}).$$

We learned that in general there is no computable $N : \mathbb{N} \to \mathbb{N}$ satisfying

$$\forall k, \forall m (|x_{N(k)+m} - x_{N(k)}| \leq 2^{-k}),$$

due to the result of Specker.

But we now have a procedure for dealing with non-computable statements like this.

Let's first take a look at a proof.

Proving the monotone convergence theorem

Proof.

Suppose that the monotone convergence principle fails i.e. there exists some k such that

$$\forall n \exists m (|x_{n+m} - x_n| > 2^{-k}).$$

Proving the monotone convergence theorem

Proof.

Suppose that the monotone convergence principle fails i.e. there exists some k such that

$$\forall n \exists m (|x_{n+m} - x_n| > 2^{-k}).$$

Then there exists a function $g : \mathbb{N} \to \mathbb{N}$ such that

$$\forall n (|x_{n+g(n)} - x_n| > 2^{-k}).$$

Proving the monotone convergence theorem

Proof.

Suppose that the monotone convergence principle fails i.e. there exists some k such that

$$\forall n \exists m (|x_{n+m} - x_n| > 2^{-k}).$$

Then there exists a function $g : \mathbb{N} \to \mathbb{N}$ such that

$$\forall n (|x_{n+g(n)} - x_n| > 2^{-k}).$$

Define the function $\tilde{g}(n) = n + g(n)$. Then we have a sequence

$$0 \leq x_0 < x_{\tilde{g}(0)} < x_{\tilde{g}(2)(0)} < \ldots <$$

with $x_{\tilde{g}^{(i+1)}(0)} - x_{\tilde{g}^{(i)}(0)} > 2^{-k}$, therefore

$$x_{\tilde{g}^{(2^k)}} > 1$$

a contradiction. □

The computational content of the proof

Our proof gave us some indirect computational information, namely

$$\forall k, g \exists n \leq \tilde{g}^{(2^k)}(0)(|x_{n+g(n)} - x_n| \leq 2^{-k}),$$

or in other words

$$\forall k, g \exists n \leq \tilde{g}^{(2^k)}(0) \forall i, j \in [n, n + g(n)](|x_i - x_j| \leq 2^{-k})$$

The computational content of the proof

Our proof gave us some indirect computational information, namely

$$\forall k, g \exists n \leq \tilde{g}^{(2^k)}(0)(|x_{n+g(n)} - x_n| \leq 2^{-k}),$$

or in other words

$$\forall k, g \exists n \leq \tilde{g}^{(2^k)}(0) \forall i, j \in [n, n+g(n)](|x_i - x_j| \leq 2^{-k})$$

Note that we can rephrase this statement entirely, so as only to refer to a finite part of (x_n). Let $M = \tilde{g}^{(2^k+1)}(0)$. We have the following:

Theorem (Finite convergence principle)

Let $k \in \mathbb{N}$, $g : \mathbb{N} \to \mathbb{N}$, and suppose that $0 \leq x_0 \leq x_1 \leq \ldots \leq x_M \leq 1$, where M is a sufficiently large number which depends only on k and g. Then there exists some $0 \leq n \leq n + g(n) \leq M$ such that $|x_i - x_j| \leq 2^{-k}$ for all $n \leq i, j \leq n + g(n)$.

This is the so-called *finite convergence principle*, made explicit by T. Tao's in

Tao, T. (2008a). Soft analysis, hard analysis, and the finite convergence principle. Essay, published as Ch. 1.3 of [Tao, 2008b], original version available online at `http://terrytao.wordpress.com/2007/05/23/soft-analysis-hard-analysis-and-the-finite-convergence-principle/`

This is the so-called *finite convergence principle*, made explicit by T. Tao's in

Tao, T. (2008a). Soft analysis, hard analysis, and the finite convergence principle. Essay, published as Ch. 1.3 of [Tao, 2008b], original version available online at `http://terrytao.wordpress.com/2007/05/23/soft-analysis-hard-analysis-and-the-finite-convergence-principle/`

- The finite convergence principle is not just an esoteric logical reformulation of a well-known concept. It is actually used in mathematics in e.g. the proof of the Szemerédi regularity lemma.

This is the so-called *finite convergence principle*, made explicit by T. Tao's in

Tao, T. (2008a). Soft analysis, hard analysis, and the finite convergence principle. Essay, published as Ch. 1.3 of [Tao, 2008b], original version available online at `http://terrytao.wordpress.com/2007/05/23/soft-analysis-hard-analysis-and-the-finite-convergence-principle/`

- The finite convergence principle is not just an esoteric logical reformulation of a well-known concept. It is actually used in mathematics in e.g. the proof of the Szemerédi regularity lemma.

- In his essay, Tao draws attention to the fact that many infinitary ('soft', qualitative') statements have finitary ('hard', 'quantitative') analogous, which have useful applications.

This is the so-called *finite convergence principle*, made explicit by T. Tao's in

Tao, T. (2008a). Soft analysis, hard analysis, and the finite convergence principle. Essay, published as Ch. 1.3 of [Tao, 2008b], original version available online at `http://terrytao.wordpress.com/2007/05/23/soft-analysis-hard-analysis-and-the-finite-convergence-principle/`

- The finite convergence principle is not just an esoteric logical reformulation of a well-known concept. It is actually used in mathematics in e.g. the proof of the Szemerédi regularity lemma.
- In his essay, Tao draws attention to the fact that many infinitary ('soft', qualitative') statements have finitary ('hard', 'quantitative') analogous, which have useful applications.
- It was later observed that this correspondence between soft and hard statements is just the classical functional interpretation!

This is the so-called *finite convergence principle*, made explicit by T. Tao's in

Tao, T. (2008a). Soft analysis, hard analysis, and the finite convergence principle. Essay, published as Ch. 1.3 of [Tao, 2008b], original version available online at `http://terrytao.wordpress.com/2007/05/23/soft-analysis-hard-analysis-and-the-finite-convergence-principle/`

- The finite convergence principle is not just an esoteric logical reformulation of a well-known concept. It is actually used in mathematics in e.g. the proof of the Szemerédi regularity lemma.

- In his essay, Tao draws attention to the fact that many infinitary ('soft', qualitative') statements have finitary ('hard', 'quantitative') analogous, which have useful applications.

- It was later observed that this correspondence between soft and hard statements is just the classical functional interpretation!

Idea. Proof interpretations do much more that just extracting numerical information. They help us understand and formalize the connection between infinitary and finintary statements in mathematics.

Convergence principles are widely studied in proof mining. Here, the functional which witnesses the corresponding finitary principle is knows as a rate of metastability.

Convergence principles are widely studied in proof mining. Here, the functional which witnesses the corresponding finitary principle is knows as a rate of metastability.

Too see the functional interpretation applied to obtain finitary versions of other infinitary principles see e.g.

Convergence principles are widely studied in proof mining. Here, the functional which witnesses the corresponding finitary principle is knows as a rate of metastability.

Too see the functional interpretation applied to obtain finitary versions of other infinitary principles see e.g.

- Gaspar, J. and Kohlenbach, U. (2010). On Tao's "finitary" infinite pigeonhole principle.
  Journal of Symbolic Logic, 75(1):355–371

Convergence principles are widely studied in proof mining. Here, the functional which witnesses the corresponding finitary principle is knows as a rate of metastability.

Too see the functional interpretation applied to obtain finitary versions of other infinitary principles see e.g.

- Gaspar, J. and Kohlenbach, U. (2010). On Tao's "finitary" infinite pigeonhole principle.
  Journal of Symbolic Logic, 75(1):355–371

- Safarik, P. and Kohlenbach, U. (2010). On the interpretation of the Bolzano-Weierstrass principle.
  Mathematical Logic Quarterly, 56(5):508–532

Convergence principles are widely studied in proof mining. Here, the functional which witnesses the corresponding finitary principle is knows as a rate of metastability.

Too see the functional interpretation applied to obtain finitary versions of other infinitary principles see e.g.

- Gaspar, J. and Kohlenbach, U. (2010). On Tao's "finitary" infinite pigeonhole principle.
  Journal of Symbolic Logic, 75(1):355–371

- Safarik, P. and Kohlenbach, U. (2010). On the interpretation of the Bolzano-Weierstrass principle.
  Mathematical Logic Quarterly, 56(5):508–532

- Powell, T. (2018). Well quasi-orders and the functional interpretation.
  To appear in: Schuster, P., Seisenberger, M. and Weiermann, A. editors, *Well Quasi-Orders in Computation, Logic, Language and Reasoning*, Trends in Logic, Springer

Outline

From function classes to arithmetics

Our initial motivation was to identify a programming language for PA.

From function classes to arithmetics

Our initial motivation was to identify a programming language for PA.

A reverse motivation: Given a programming language, or complexity class, develop an appropriate arithmetic.

From function classes to arithmetics

Our initial motivation was to identify a programming language for PA.

A reverse motivation: Given a programming language, or complexity class, develop an appropriate arithmetic.

Write $\mathsf{I}\Sigma_1$ for the fragment of PA that only allows induction on $\exists$ formulae (*i.e.*, *semi-recursive* predicates). The following is a seminal result:

Theorem (C. Parsons '72)

If $\mathsf{I}\Sigma_1 \vdash \forall x.\exists! y.Q(x,y)$, *then* $Q(x,y)$ *is the graph of a primitive recursive function.*

From function classes to arithmetics

Our initial motivation was to identify a programming language for PA.

A reverse motivation: Given a programming language, or complexity class, develop an appropriate arithmetic.

Write $\mathsf{I}\Sigma_1$ for the fragment of PA that only allows induction on $\exists$ formulae (*i.e.*, *semi-recursive* predicates). The following is a seminal result:

Theorem (C. Parsons '72)

If $\mathsf{I}\Sigma_1 \vdash \forall x.\exists! y.Q(x,y)$, *then* $Q(x,y)$ *is the graph of a primitive recursive function.*

This is proved by using a restriction $\hat{\mathsf{T}}$ of system T to predicative recursors:

$$\begin{aligned} \hat{R}(f,g,0,b) &= f(b) \\ \hat{R}(f,g,\mathsf{s}n,b) &= g(n,\hat{R}(f,g,n,b),b) \end{aligned}$$

Indeed, this system allows us to recover a *bona fide* interpretation $\mathsf{I}\Sigma_1 \to \hat{\mathsf{T}}$ that satisfies all the usual properties we would expect.

From function classes to arithmetics

Our initial motivation was to identify a programming language for PA.

A reverse motivation: Given a programming language, or complexity class, develop an appropriate arithmetic.

Write $\mathsf{I}\Sigma_1$ for the fragment of PA that only allows induction on $\exists$ formulae (*i.e.*, *semi-recursive* predicates). The following is a seminal result:

Theorem (C. Parsons '72)

If $\mathsf{I}\Sigma_1 \vdash \forall x.\exists! y.Q(x,y)$, *then* $Q(x,y)$ *is the graph of a primitive recursive function.*

This is proved by using a restriction $\hat{\mathsf{T}}$ of system T to predicative recursors:

$$\begin{aligned} \hat{R}(f,g,0,b) &= f(b) \\ \hat{R}(f,g,\mathsf{s}n,b) &= g(n,\hat{R}(f,g,n,b),b) \end{aligned}$$

Indeed, this system allows us to recover a *bona fide* interpretation $\mathsf{I}\Sigma_1 \to \hat{\mathsf{T}}$ that satisfies all the usual properties we would expect.

Furthermore $\hat{\mathsf{T}}$ may naturally be interpreted, for appropriate theorems, into a quantifier-free classical system PRA over the primitive recursive functions.

An alternative to Dialectica: the WFM

As nice as Parsons' result is, the functional approach seems like unintuitively large machinery for such a concrete programming language.

An alternative to Dialectica: the WFM

As nice as Parsons' result is, the functional approach seems like unintuitively large machinery for such a concrete programming language.

In fact, there is another way! It is known as the **Witness Function Method**, due to G. Mints and S. Buss.

Some important *differences*:

- Works directly in classical logic.
- Only concrete functions on $\mathbb{N}$ are used. No need for functionals/higher types!

An alternative to Dialectica: the WFM

As nice as Parsons' result is, the functional approach seems like unintuitively large machinery for such a concrete programming language.

In fact, there is another way! It is known as the **Witness Function Method**, due to G. Mints and S. Buss.

Some important *differences*:

- Works directly in classical logic.
- Only concrete functions on $\mathbb{N}$ are used. No need for functionals/higher types!

Nonetheless, there are certain similarities in the structure of the proofs. The following play similar roles:

N-int. to negative fragment	De Morgan trans. to $\neg$-free fragment
D-int. to system $\hat{\mathsf{T}}$	Witnessing to PRA

An alternative to Dialectica: the WFM

As nice as Parsons' result is, the functional approach seems like unintuitively large machinery for such a concrete programming language.

In fact, there is another way! It is known as the **Witness Function Method**, due to G. Mints and S. Buss.

Some important *differences*:

- Works directly in classical logic.
- Only concrete functions on $\mathbb{N}$ are used. No need for functionals/higher types!

Nonetheless, there are certain similarities in the structure of the proofs. The following play similar roles:

N-int. to negative fragment	De Morgan trans. to $\neg$-free fragment
D-int. to system $\hat{\mathsf{T}}$	Witnessing to PRA

NB: However the engines behind these results are rather different. In particular the witnessing is only possible in classical logic thanks to the cut-elimination theorem. This result, at the same time, 'includes' the work required in translating $\hat{\mathsf{T}}$ to PRA.

Cut-elimination

Cut-elimination

G. Gentzen was the founder of structural proof theory, and used low-level tools to contribute a range of tools to mathematical logic.

His **sequent calculus** is still the most important proof system in proof theory today.

Cut-elimination

G. Gentzen was the founder of structural proof theory, and used low-level tools to contribute a range of tools to mathematical logic.

His **sequent calculus** is still the most important proof system in proof theory today.

The basic object of reasoning is a **sequent**:

$$A_1, \ldots, A_m \Rightarrow B_1, \ldots, B_n \quad \text{interpreted as} \quad \bigwedge_{i=1}^{m} A_i \Rightarrow \bigvee_{j=1}^{n} B_j$$

Cut-elimination

G. Gentzen was the founder of structural proof theory, and used low-level tools to contribute a range of tools to mathematical logic.

His **sequent calculus** is still the most important proof system in proof theory today.

The basic object of reasoning is a **sequent**:

$$A_1, \ldots, A_m \Rightarrow B_1, \ldots, B_n \quad \text{interpreted as} \quad \bigwedge_{i=1}^{m} A_i \Rightarrow \bigvee_{j=1}^{n} B_j$$

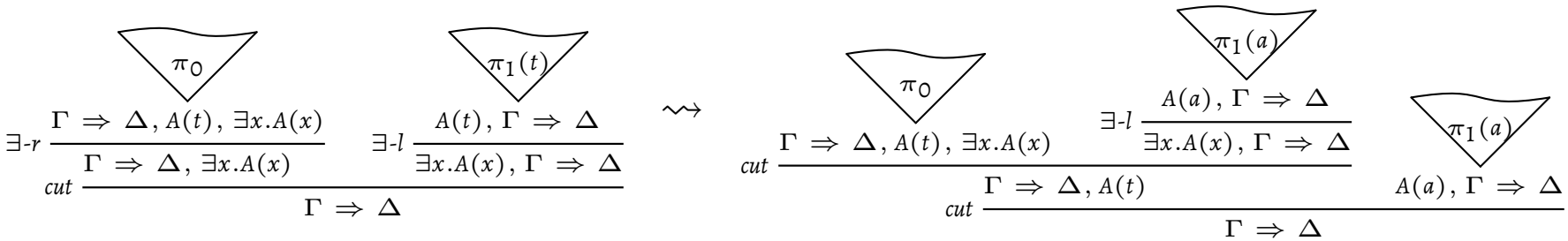
Hauptsatz (G. Gentzen, 1934)

For any first-order proof in the sequent calculus, there is one of the same conclusion s.t. all formulae occurring are subformulae of the conclusion, up to substitution of terms for variables.

NB: This is one of the deepest results in mathematical logic!

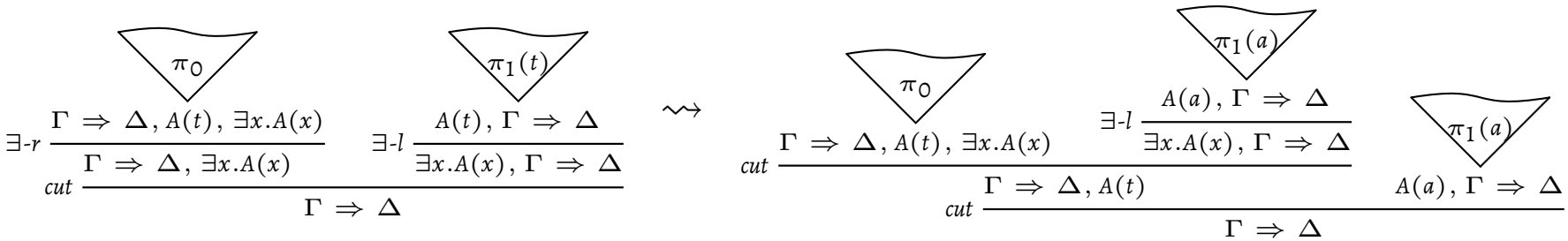
A glimpse of cut-reduction

A prototypical cut-reduction case:



A glimpse of cut-reduction

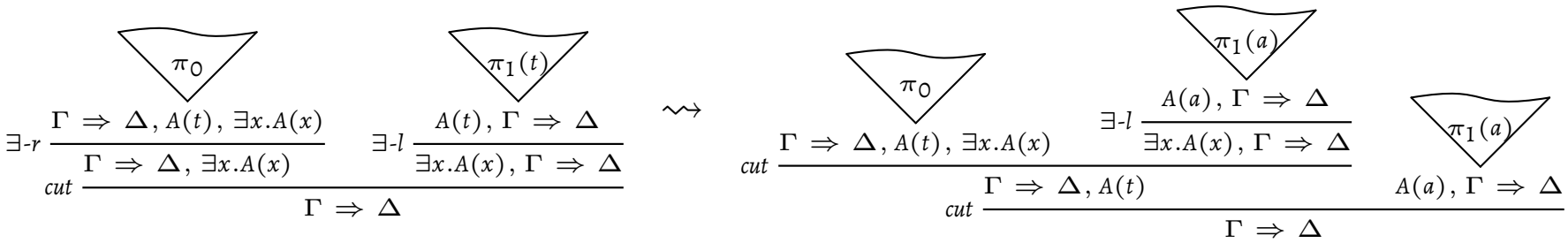
A prototypical cut-reduction case:



- Proved using a sophisticated induction measure.
- Non-elementary complexity.
- Closely related to computation via the *Curry-Howard correspondence*.

A glimpse of cut-reduction

A prototypical cut-reduction case:



- Proved using a sophisticated induction measure.
- Non-elementary complexity.
- Closely related to computation via the *Curry-Howard correspondence*.

Here we rather use it for the following normal form property:

Corollary

In the case of arithmetic, any PA *proof can be written as a sequent-style proof where all formulae are subformulae of the conclusion, an induction formula or an axiom, up to…*

A rudimentary system for $I\Sigma_1$

For convenience, we may assume that all primitive recursive functions may occur as terms of $I\Sigma_1$. (Indeed, $I\Sigma_1$ can already prove that such functions are well-defined.)

A rudimentary system for $\mathrm{I}\Sigma_1$

For convenience, we may assume that all primitive recursive functions may occur as terms of $\mathrm{I}\Sigma_1$. (Indeed, $\mathrm{I}\Sigma_1$ can already prove that such functions are well-defined.)

Instead of the N-interpretation, we have De Morgan normalisation of formulae and proofs:

Negation elimination

Every proof may be written in $\neg$-free form, via the following local translation on formulae:

$$\begin{array}{rcl} (\neg\, s \leq t)^n & := & t < s \\ (\neg(A \vee B))^n & := & \neg A^n \wedge \neg B^n \\ (\neg(A \wedge B))^n & := & \neg A^n \vee \neg B^n \end{array} \qquad \begin{array}{rcl} (\neg\exists x.A)^n & := & \forall x.\neg A^n \\ (\neg\forall x.A)^n & := & \exists x.\neg A^n \end{array}$$

No need for implication: this is classical logic!

A rudimentary system for $\mathrm{I}\Sigma_1$

For convenience, we may assume that all primitive recursive functions may occur as terms of $\mathrm{I}\Sigma_1$. (Indeed, $\mathrm{I}\Sigma_1$ can already prove that such functions are well-defined.)

Instead of the N-interpretation, we have De Morgan normalisation of formulae and proofs:

Negation elimination

Every proof may be written in $\neg$-free form, via the following local translation on formulae:

$$\begin{array}{rcl} (\neg\, s \leq t)^n & := & t < s \\ (\neg(A \vee B))^n & := & \neg A^n \wedge \neg B^n \\ (\neg(A \wedge B))^n & := & \neg A^n \vee \neg B^n \end{array} \qquad \begin{array}{rcl} (\neg\exists x.A)^n & := & \forall x.\neg A^n \\ (\neg\forall x.A)^n & := & \exists x.\neg A^n \end{array}$$

No need for implication: this is classical logic!
Along with cut-elimination we have:

Normalisation

Any $\mathrm{I}\Sigma_1$ proof of $\forall x.\exists y.Q(x,y)$ sentence can be translated into an arithmetic sequent proof of $\Rightarrow \exists y.Q(a,y)$ consisting of only $\exists$ formulae.

Interpretation into primitive recursive arithmetic.

PRA is something like system T restricted to only type 0 recursors. Equivalently, it is a quantifier-free version of PA equipped with terms for all primitive recursive functions. It is much simpler!

Interpretation into primitive recursive arithmetic.

PRA is something like system T restricted to only type 0 recursors. Equivalently, it is a quantifier-free version of PA equipped with terms for all primitive recursive functions. It is much simpler!

For $\neg$-free formulae, we may reuse a modest variation of the D-interpretation (called d here).

Interpretation into primitive recursive arithmetic.

PRA is something like system T restricted to only type 0 recursors. Equivalently, it is a quantifier-free version of PA equipped with terms for all primitive recursive functions. It is much simpler!

For $\neg$-free formulae, we may reuse a modest variation of the D-interpretation (called d here). However, it is the crucial case of implication where the treatment differs.

$$\exists x.A(x,c) \Rightarrow \exists y.B(y,c) \qquad \mapsto \qquad \exists y.(A_d(a,c) \to B_d(y,c))$$

for some primitive recursive term t.

Interpretation into primitive recursive arithmetic.

PRA is something like system T restricted to only type 0 recursors. Equivalently, it is a quantifier-free version of PA equipped with terms for all primitive recursive functions. It is much simpler!

For $\neg$-free formulae, we may reuse a modest variation of the D-interpretation (called d here). However, it is the crucial case of implication where the treatment differs.

$$\exists x.A(x,c) \Rightarrow \exists y.B(y,c) \qquad \mapsto \qquad \exists y.(A_d(a,c) \to B_d(y,c))$$

for some primitive recursive term t.

Here $\to$ is material implication: we remain in a classical theory.

Interpretation into primitive recursive arithmetic.

PRA is something like system T restricted to only type 0 recursors. Equivalently, it is a quantifier-free version of PA equipped with terms for all primitive recursive functions. It is much simpler!

For $\neg$-free formulae, we may reuse a modest variation of the D-interpretation (called d here). However, it is the crucial case of implication where the treatment differs.

$$\exists x.A(x,c) \Rightarrow \exists y.B(y,c) \qquad \mapsto \qquad \exists y.(A_d(a,c) \to B_d(y,c))$$

for some primitive recursive term t.

Here $\to$ is material implication: we remain in a classical theory.

Aside: We typically have to deal with formulae like $\forall x < t.\exists y.A(x,y,c)$. Again, the treatment of $\forall$ here differs from Dialectica, thanks to the **collection principle**:

$$\forall x < t.\exists y.A(x,y,c) \Rightarrow \exists z.\forall x < t.\exists y < z.A(x,y,c)$$

Interpretation of $\forall\exists$ theorems

Theorem (G. Mints '73, S. Buss '95)

For any $\mathsf{I}\Sigma_1$ *sequent proof of a sequent* $S(a)$ *there is a* PRA *proof of* $S_D(a, t(a))$ *for some appropriate term* t *of* PRA.

Interpretation of $\forall\exists$ theorems

Theorem (G. Mints '73, S. Buss '95)

For any $\mathsf{I}\Sigma_1$ *sequent proof of a sequent* $S(a)$ *there is a* PRA *proof of* $S_D(a, t(a))$ *for some appropriate term* t *of* PRA.

Some of the key cases closely mirror those of Dialectica, e.g.:

$$\exists x.A(x,c) \vee \exists x.A(x,c) \Rightarrow \exists x.A(x,c) \quad \mapsto \quad A_d(a,c) \vee A_d(b,c) \to A_d(t(a,b,c),c)$$

where $t(a,b,c) = \texttt{if } A_D(a,c) \texttt{ is odd then } a \texttt{ else } b$.

Interpretation of $\forall\exists$ theorems

Theorem (G. Mints '73, S. Buss '95)

For any $\mathsf{I}\Sigma_1$ *sequent proof of a sequent* $S(a)$ *there is a* PRA *proof of* $S_D(a, t(a))$ *for some appropriate term* t *of* PRA.

Some of the key cases closely mirror those of Dialectica, e.g.:

$$\exists x.A(x,c) \vee \exists x.A(x,c) \Rightarrow \exists x.A(x,c) \quad \mapsto \quad A_d(a,c) \vee A_d(b,c) \to A_d(t(a,b,c),c)$$

where $t(a,b,c) = \texttt{if } A_D(a,c) \texttt{ is odd then } a \texttt{ else } b$.

However, as well as implication, the $\forall$ also behaves differently:

$$\forall \frac{a < t(c) \Rightarrow \exists y.Q(a,y,c)}{\Rightarrow \exists z.\forall x < t(c).\exists y < z.Q(x,y,c)} \quad \mapsto \quad \frac{a < t(c) \Rightarrow Q(a,u(a,c),c)}{\Rightarrow \forall x < t(c).\exists y < v(c).Q(x,v(x,c),c)}$$

where $v(c) := \max\{u(a,c) : 0 < a < t(c)\}$.

Interpretation of $\forall\exists$ theorems

Theorem (G. Mints '73, S. Buss '95)

For any $\mathsf{I}\Sigma_1$ *sequent proof of a sequent* $S(a)$ *there is a* PRA *proof of* $S_D(a, t(a))$ *for some appropriate term* t *of* PRA.

Some of the key cases closely mirror those of Dialectica, e.g.:

$$\exists x.A(x,c) \vee \exists x.A(x,c) \Rightarrow \exists x.A(x,c) \quad \mapsto \quad A_d(a,c) \vee A_d(b,c) \to A_d(t(a,b,c),c)$$

where $t(a,b,c) = \texttt{if } A_D(a,c) \texttt{ is odd then } a \texttt{ else } b$.

However, as well as implication, the $\forall$ also behaves differently:

$$\forall \frac{a < t(c) \Rightarrow \exists y.Q(a,y,c)}{\Rightarrow \exists z.\forall x < t(c).\exists y < z.Q(x,y,c)} \quad \mapsto \quad \frac{a < t(c) \Rightarrow Q(a,u(a,c),c)}{\Rightarrow \forall x < t(c).\exists y < v(c).Q(x,v(x,c),c)}$$

where $v(c) := \max\{u(a,c) : 0 < a < t(c)\}$.

Corollary

The $\mathsf{I}\Sigma_1 \vdash \forall x.\exists! y.Q(x,y)$ *just if* $Q(x,y)$ *is the graph of a primitive recursive function.*

Perspectives

Perspectives

Motto

Induction on semi-recursive predicates characterises primitive recursion.

Perspectives

Motto

Induction on semi-recursive predicates characterises primitive recursion.

A complementary approach to Parsons result. Exemplifies the trade-off of recursion on higher types vs. recursion on more sophisticated well-orders.

The method is much simpler and more intuitive.

Perspectives

Motto

Induction on semi-recursive predicates characterises primitive recursion.

A complementary approach to Parsons result. Exemplifies the trade-off of recursion on higher types vs. recursion on more sophisticated well-orders.

The method is much simpler and more intuitive.

The approach has been extended to all of PA by S. Buss by extending by appealing to recursion on more complex orders.

Perspectives

Motto

Induction on semi-recursive predicates characterises primitive recursion.

A complementary approach to Parsons result. Exemplifies the trade-off of recursion on higher types vs. recursion on more sophisticated well-orders.

The method is much simpler and more intuitive.

The approach has been extended to all of PA by S. Buss by extending by appealing to recursion on more complex orders.

However, the real advantage is for reasoning about weak theories, as we will see. In contrast the Dialectica interpretation is better for strong theories.

Key references

- Parsons, C. (1972). On *n*-quantifier induction.
  J. Symbolic Logic, 37(3):466–482

- Buss, S. R. (1991). The witness function method and provably recursive functions of peano arithmetic.
  In *Logic, Methodology and Philosophy of Science IX*, pages 29–68. North Holland

- van Heijenoort, J. (1972). The collected papers of gerhard gentzen. m. e. szabo.
  Philosophy of Science, 39(1):91–91

Outline

From logic to complexity

The WFM is useful for theories for reasoning about computational complexity and computability.

From logic to complexity

The WFM is useful for theories for reasoning about computational complexity and computability.

There is a whole world of theories underneath $\mathsf{I}\Sigma_1$ and classes underneath primitive recursion! Again, the fundamental idea is to restrict induction invariants, and sometimes other logical and arithmetical principles.

From logic to complexity

The WFM is useful for theories for reasoning about computational complexity and computability.

There is a whole world of theories underneath $\mathsf{I}\Sigma_1$ and classes underneath primitive recursion! Again, the fundamental idea is to restrict induction invariants, and sometimes other logical and arithmetical principles.

Bounded arithmetic has been an immensely successful approach towards characterising *feasible* complexity classes.

From logic to complexity

The WFM is useful for theories for reasoning about computational complexity and computability.

There is a whole world of theories underneath $\mathsf{I}\Sigma_1$ and classes underneath primitive recursion! Again, the fundamental idea is to restrict induction invariants, and sometimes other logical and arithmetical principles.

Bounded arithmetic has been an immensely successful approach towards characterising *feasible* complexity classes. To date there exist characterisations of:

- Polynomial time (**P**) and each level of the polynomial hierarchy (**PH**).
- Feasible parallel time ($\mathbf{NC}^1$) and the $\mathbf{NC}^i$-$\mathbf{AC}^i$ hierarchy.
- Logarithmic space (**L**) and its (co-)nondeterministic versions ($\mathbf{NL} = \mathbf{coNL}$).
- Polynomial space (**PSPACE**), exponential time (**EXP**), ...

Key references

- Clote, P. and Kranakis, E. (2002). *Boolean Functions and Computation Models*
- Buss, S. R. (1986). *Bounded arithmetic.*
  PhD thesis
- Krajicek, J. (1995). *Bounded Arithmetic, Propositional Logic and Complexity Theory.*
  Encyclopedia of Mathematics and its Applications. Cambridge University Press
- Cook, S. and Nguyen, P. (2010). *Logical Foundations of Proof Complexity.*
  Cambridge University Press, New York, NY, USA, 1st edition

The case of polynomial-time

Remember the motto:

Induction on semi-recursive predicates characterises primitive recursion.

The case of polynomial-time

Remember the motto:

Induction on semi-recursive predicates characterises primitive recursion.

It turns out there is an elegant *feasible* version of this:

Induction on **NP** *predicates characterises polynomial time.*

The case of polynomial-time

Remember the motto:

Induction on semi-recursive predicates characterises primitive recursion.

It turns out there is an elegant *feasible* version of this:

Induction on **NP** *predicates characterises polynomial time.*

For this we need a programming language for polynomial time!

Bounded recursion on notation (A. Cobham, 1964)

We say that f is defined by **bounded recursion on notation** from functions g, h, k if,

$$\begin{aligned} f(0) &= g \\ f(2x) &= h_0(x, f(x)) \quad \text{if } x \neq 0 \\ f(2x+1) &= h_1(x, f(x)) \end{aligned}$$

and $f(x) \leq k(x)$ for all x.

The case of polynomial-time

Remember the motto:

Induction on semi-recursive predicates characterises primitive recursion.

It turns out there is an elegant *feasible* version of this:

Induction on **NP** *predicates characterises polynomial time.*

For this we need a programming language for polynomial time!

Bounded recursion on notation (A. Cobham, 1964)

We say that f is defined by **bounded recursion on notation** from functions g, h, k if,

$$\begin{aligned} f(0) &= g \\ f(2x) &= h_0(x, f(x)) \quad \text{if } x \neq 0 \\ f(2x+1) &= h_1(x, f(x)) \end{aligned}$$

and $f(x) \leq k(x)$ for all x.

Polynomial-time completeness: *The class of primitive recursive functions that only use recursion of the form above, plus the function $x^{\log y}$, are complete for polynomial time.*

Seminal results

We may define a theory S_2^1 that allows induction only on formulae of the form $\exists x < t.Q(x, a)$, analogous to $\mathsf{I}\Sigma_1$.

We may also define a quantifier-free theory PV based on bounded recursion on notation, analogous to PRA.

Seminal results

We may define a theory S_2^1 that allows induction only on formulae of the form $\exists x < t.Q(x, a)$, analogous to $\mathsf{I}\Sigma_1$.

We may also define a quantifier-free theory PV based on bounded recursion on notation, analogous to PRA.

Theorem (S. Buss, 1985, informally)

The $\forall\exists$ fragment of S_2^1 is interpreted by the WFM into PV. *In particular, if $S_2^1 \vdash \forall x.\exists!y.Q(x, y)$ then $Q(x, y)$ is the graph of a polynomial-time function.*

The case of polynomial time is particularly interesting since it serves as a litmus test for comparing different proof interpretations.

Theorem (S. Cook & A. Urquhart, 1993)

All of S_2^1 is ND-interpreted into a higher type version of PV.

Seminal results

We may define a theory S_2^1 that allows induction only on formulae of the form $\exists x < t.Q(x, a)$, analogous to $\mathsf{I}\Sigma_1$.

We may also define a quantifier-free theory PV based on bounded recursion on notation, analogous to PRA.

Theorem (S. Buss, 1985, informally)

The $\forall\exists$ fragment of S_2^1 is interpreted by the WFM into PV. *In particular, if $S_2^1 \vdash \forall x.\exists!y.Q(x, y)$ then $Q(x, y)$ is the graph of a polynomial-time function.*

The case of polynomial time is particularly interesting since it serves as a litmus test for comparing different proof interpretations.

Theorem (S. Cook & A. Urquhart, 1993)

All of S_2^1 is ND-interpreted into a higher type version of PV.

Corollary (alternative to Buss, '85)

If intuitionistic-$S_2^1 \vdash \forall x.\exists y.A(x, y)$ for arbitrary A, then there is a term $t(x)$ of PV *such that intuitionistic-$S_2^1 \vdash \forall x.A(x, f(x))$.*

Applications to proof complexity

- Proof interpretations intimately connected to the size of proofs in propositional proof systems.
  (see, e.g., [Krajicek, 1995]).
- Bounded reverse mathematics: characterising the nondeterministic complexity of results in combinatorics and complexity.
  (see, e.g., [Cook and Nguyen, 2010])
- Elegant interplays between the WFM and realisability-style interpretations.
  [Das, 2016]
  Work in progress: proof complexity manipulations from the Dialectica interpretation.

Getting rid of bounds: implicit complexity

- Recursion-theoretic characterisations of complexity classes with no explicit resource bounds!
  (see, e.g., [Bellantoni and Cook, 1992]).
- Theories for these based on *ramification*, or *modalities*.
  (see, e.g., [Leivant, 1994])
- Resource sensitive logics + WFM = entirely implicit characterisations.
  [Baillot and Das, 2016]

Outline

The initial soundness theorem for the functional interpretation interprets Peano arithmetic in system T:

$$\mathrm{PA}^{\omega} \mapsto \mathrm{T}$$

Roughly speaking, PA^{ω} is just logic + induction, and the core of the soundness theorem is the following correspondence:

$$\text{induction} \mapsto \text{primitive recursion in all finite types}$$

The initial soundness theorem for the functional interpretation interprets Peano arithmetic in system T:

$$\mathrm{PA}^{\omega} \mapsto \mathrm{T}$$

Roughly speaking, PA^{ω} is just logic + induction, and the core of the soundness theorem is the following correspondence:

$$\text{induction} \mapsto \text{primitive recursion in all finite types}$$

But many mathematical proofs go beyond Peano arithmetic, and in particular require some stronger principles such as comprehension, the axiom of choice, Zorn's lemma.

This gives rise to the following general problem:

$$\text{strong principle} \mapsto ???$$

Let's look at a simple version of the axiom of choice. Consider the drinkers paradox from the last lecture, now with an additional parameter n:

$$\forall n \exists x (P(n,x) \to \forall y P(n,y))$$

Then we can infer (using the axiom of choice) that there exists some function $f : \mathbb{N} \to X$ such that

$$\forall n (P(n, f(n)) \to \forall y P(n,y)).$$

> *For any countable set of pubs there is a countable set of drinkers such that if drinker n is drinking, then everyone in pub n is drinking.*

Let's look at a simple version of the axiom of choice. Consider the drinkers paradox from the last lecture, now with an additional parameter n:

$$\forall n \exists x (P(n,x) \to \forall y P(n,y))$$

Then we can infer (using the axiom of choice) that there exists some function $f : \mathbb{N} \to X$ such that

$$\forall n (P(n, f(n)) \to \forall y P(n,y)).$$

> *For any countable set of pubs there is a countable set of drinkers such that if drinker n is drinking, then everyone in pub n is drinking.*

We already know how to solve the (classical) functional interpretation of ordinary drinkers paradox for pub n:

$$\forall n, g \exists x (P(n,x) \to P(n, g(x)))$$

namely via the functional

$$\Phi(n,g) := \begin{cases} 0 & \text{if } P(n, g(0)) \\ g(0) & \text{if } \neg P(n, g(0)) \end{cases}$$

Now consider the functional interpretation of our 'function' form of the drinkers paradox

$$\exists f \forall n (P(n, f(n)) \to \forall y P(n, y)).$$

Which is

$$\forall \omega, \varphi \exists f (P(\omega f, f(\omega f)) \to P(\omega f, \varphi f))$$

Now consider the functional interpretation of our 'function' form of the drinkers paradox

$$\exists f \forall n (P(n, f(n)) \to \forall y P(n, y)).$$

Which is

$$\forall \omega, \varphi \exists f (P(\omega f, f(\omega f)) \to P(\omega f, \varphi f))$$

Let $a : \mathbb{N} \to X$ be a partial function and let $\hat{a}$ be the extension of this function with some canonical element of type X. Define

$$\Psi(\omega, \varphi)(a) := \begin{cases} \hat{a} & \text{if } \omega(\hat{a}) \in \mathrm{dom}(a) \text{ or } P(\omega(\hat{a}), \varphi(\hat{a})) \\ \Psi(\omega, \varphi)(a[\omega(\hat{a}) \mapsto \varphi(\hat{a})]) & \text{otherwise.} \end{cases}$$

This is a new form of recursion over *extensions of partial functions*.

Now consider the functional interpretation of our 'function' form of the drinkers paradox

$$\exists f \forall n (P(n, f(n)) \to \forall y P(n, y)).$$

Which is

$$\forall \omega, \varphi \exists f (P(\omega f, f(\omega f)) \to P(\omega f, \varphi f))$$

Let $a : \mathbb{N} \to X$ be a partial function and let $\hat{a}$ be the extension of this function with some canonical element of type X. Define

$$\Psi(\omega, \varphi)(a) := \begin{cases} \hat{a} & \text{if } \omega(\hat{a}) \in \mathrm{dom}(a) \text{ or } P(\omega(\hat{a}), \varphi(\hat{a})) \\ \Psi(\omega, \varphi)(a[\omega(\hat{a}) \mapsto \varphi(\hat{a})]) & \text{otherwise.} \end{cases}$$

This is a new form of recursion over *extensions of partial functions*.

It is actually a weak form of bar recursion. The study of bar recursive extensions of System T and their relationship to choice/comprehension has been an active area of research in the last decade or so.

Extensions of proof interpretations and strong forms of recursion

Extensions of proof interpretations and strong forms of recursion

- Spector, C. (1962). Provably recursive functionals of analysis: a consistency proof of analysis by an extension of principles in current intuitionistic mathematics.
  In Dekker, F. D. E., editor, *Recursive Function Theory: Proc. Symposia in Pure Mathematics*, volume 5, pages 1–27. American Mathematical Society, Providence, Rhode Island

Extensions of proof interpretations and strong forms of recursion

- Spector, C. (1962). Provably recursive functionals of analysis: a consistency proof of analysis by an extension of principles in current intuitionistic mathematics.
  In Dekker, F. D. E., editor, *Recursive Function Theory: Proc. Symposia in Pure Mathematics*, volume 5, pages 1–27. American Mathematical Society, Providence, Rhode Island
- Burr, W. (1998). *Functionals in Set Theory and Arithmetic.*
  PhD thesis, Westfälische Wilhelms-Universität Münster

Extensions of proof interpretations and strong forms of recursion

- Spector, C. (1962). Provably recursive functionals of analysis: a consistency proof of analysis by an extension of principles in current intuitionistic mathematics.
  In Dekker, F. D. E., editor, *Recursive Function Theory: Proc. Symposia in Pure Mathematics*, volume 5, pages 1–27. American Mathematical Society, Providence, Rhode Island
- Burr, W. (1998). *Functionals in Set Theory and Arithmetic.*
  PhD thesis, Westfälische Wilhelms-Universität Münster
- Escardó, M. and Oliva, P. (2011). Sequential games and optimal strategies.
  Royal Society Proceedings A, 467:1519–1545

Extensions of proof interpretations and strong forms of recursion

- Spector, C. (1962). Provably recursive functionals of analysis: a consistency proof of analysis by an extension of principles in current intuitionistic mathematics.
  In Dekker, F. D. E., editor, *Recursive Function Theory: Proc. Symposia in Pure Mathematics*, volume 5, pages 1–27. American Mathematical Society, Providence, Rhode Island

- Burr, W. (1998). *Functionals in Set Theory and Arithmetic.*
  PhD thesis, Westfälische Wilhelms-Universität Münster

- Escardó, M. and Oliva, P. (2011). Sequential games and optimal strategies.
  Royal Society Proceedings A, 467:1519–1545

- van den Berg, B., Briseid, E., and Safarik, P. (2012). A functional interpretation for nonstandard arithmetic.
  Annals of Pure and Applied Logic, 163(12):1962–1994

Extensions of proof interpretations and strong forms of recursion

- Spector, C. (1962). Provably recursive functionals of analysis: a consistency proof of analysis by an extension of principles in current intuitionistic mathematics.
  In Dekker, F. D. E., editor, *Recursive Function Theory: Proc. Symposia in Pure Mathematics*, volume 5, pages 1–27. American Mathematical Society, Providence, Rhode Island
- Burr, W. (1998). *Functionals in Set Theory and Arithmetic.*
  PhD thesis, Westfälische Wilhelms-Universität Münster
- Escardó, M. and Oliva, P. (2011). Sequential games and optimal strategies.
  Royal Society Proceedings A, 467:1519–1545
- van den Berg, B., Briseid, E., and Safarik, P. (2012). A functional interpretation for nonstandard arithmetic.
  Annals of Pure and Applied Logic, 163(12):1962–1994
- Powell, T. (2013). *On Bar Recursive Interpretations of Analysis.*
  PhD thesis, Queen Mary University of London

Extensions of proof interpretations and strong forms of recursion

- Spector, C. (1962). Provably recursive functionals of analysis: a consistency proof of analysis by an extension of principles in current intuitionistic mathematics.
  In Dekker, F. D. E., editor, *Recursive Function Theory: Proc. Symposia in Pure Mathematics*, volume 5, pages 1–27. American Mathematical Society, Providence, Rhode Island
- Burr, W. (1998). *Functionals in Set Theory and Arithmetic.*
  PhD thesis, Westfälische Wilhelms-Universität Münster
- Escardó, M. and Oliva, P. (2011). Sequential games and optimal strategies.
  Royal Society Proceedings A, 467:1519–1545
- van den Berg, B., Briseid, E., and Safarik, P. (2012). A functional interpretation for nonstandard arithmetic.
  Annals of Pure and Applied Logic, 163(12):1962–1994
- Powell, T. (2013). *On Bar Recursive Interpretations of Analysis.*
  PhD thesis, Queen Mary University of London
- Sanders, S. (2018). The Gandy-Hyland functional and a computational aspect of nonstandard analysis.
  Computability, 7(1):7–43

Outline

In lecture 3 we 'extracted' a list reversing program. Our input was a proof that

$$\forall a \exists b \mathrm{Rev}(a, b)$$

and the result was a term t of System T (i.e. a program) satisfying

$$\forall a \mathrm{Rev}(a, t(a)).$$

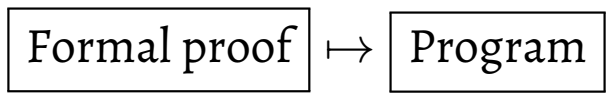
In lecture 3 we 'extracted' a list reversing program. Our input was a proof that

$$\forall a \exists b \mathrm{Rev}(a, b)$$

and the result was a term t of System T (i.e. a program) satisfying

$$\forall a \mathrm{Rev}(a, t(a)).$$

Can we automate proof interpretations i.e. write a piece of software which transforms a formal proof to a program:



In lecture 3 we 'extracted' a list reversing program. Our input was a proof that

$$\forall a \exists b \mathrm{Rev}(a, b)$$

and the result was a term t of System T (i.e. a program) satisfying

$$\forall a \mathrm{Rev}(a, t(a)).$$

Can we automate proof interpretations i.e. write a piece of software which transforms a formal proof to a program:

$$\boxed{\text{Formal proof}} \mapsto \boxed{\text{Program}}$$

Again, the answer is YES! There is even a proof assistant - MINLOG - dedicated to program extraction via functional interpretations:

http://www.mathematik.uni-muenchen.de/~logik/minlog/

What do we get out of formal extraction?

A program is written to satisfy a given specification:

$$\forall x \exists y S(x, y)$$

where $S(x, y)$ specifies how the input should be related to the output.

What do we get out of formal extraction?

A program is written to satisfy a given specification:

$$\forall x \exists y S(x, y)$$

where $S(x, y)$ specifies how the input should be related to the output.

What a programmer might do.

- Write a program satisfying the specification.
- Debug until they are convinced that it works as it should.

What do we get out of formal extraction?

A program is written to satisfy a given specification:

$$\forall x \exists y S(x, y)$$

where $S(x, y)$ specifies how the input should be related to the output.

What a programmer might do.

- Write a program satisfying the specification.
- Debug until they are convinced that it works as it should.

What a proof theorist might do.

- Write a formal proof that for any input, an output satisfying the specification exists.
- Push a button and extract a program.
- No debugging required!
- We may even get additional information, such as a bound on its complexity.

It's not quite as easy as that...

Sounds great! Why aren't programmers using proof interpretations?

It's not quite as easy as that...

Sounds great! Why aren't programmers using proof interpretations?

- *"Is your list sorting program as good as one a human would write?"*
- *"We use C++. What good to us is a program written in System T?"*
- *"Your extracted program takes up ten pages of text. How does it even work?"*
- *"Group X can already do formal verification and have developed an extremely successful tool."*
- *"Could your technique for synthesising programs be easily used by someone working at the Guardian newspaper?"*

These are all valid points...

Obstacles to overcome

... which highlight the following problems:

Obstacles to overcome

... which highlight the following problems:

Efficiency: It's easy to extract a brute force algorithm, but much more difficult to produce something intelligent, comparable to what a human would write.

Obstacles to overcome

... which highlight the following problems:

Efficiency: It's easy to extract a brute force algorithm, but much more difficult to produce something intelligent, comparable to what a human would write.

Language: Formally extracted programs are typically presented in an abstract language like System T. Real programming languages tend to follow a completely different paradigm, with concepts such as global state, concurrency, and so on...

Obstacles to overcome

... which highlight the following problems:

Efficiency: It's easy to extract a brute force algorithm, but much more difficult to produce something intelligent, comparable to what a human would write.

Language: Formally extracted programs are typically presented in an abstract language like System T. Real programming languages tend to follow a completely different paradigm, with concepts such as global state, concurrency, and so on...

Scale: Formal verification is a huge business and lots of sophisticated tools have already been developed. On top of this, most big proof assistants (Coq, NUPRL, etc) can extract programs from proofs. A small community in proof theory, dedicated to a particular style of program extraction, cannot possibly compete with this directly.

Obstacles to overcome

... which highlight the following problems:

Efficiency: It's easy to extract a brute force algorithm, but much more difficult to produce something intelligent, comparable to what a human would write.

Language: Formally extracted programs are typically presented in an abstract language like System T. Real programming languages tend to follow a completely different paradigm, with concepts such as global state, concurrency, and so on...

Scale: Formal verification is a huge business and lots of sophisticated tools have already been developed. On top of this, most big proof assistants (Coq, NUPRL, etc) can extract programs from proofs. A small community in proof theory, dedicated to a particular style of program extraction, cannot possibly compete with this directly.

Accessibility: Ultimately, methods for synthesising verified programs are only useful if they can be used by a non-specialist.

Some first steps

The synthesis of verified programs using proof interpretations like the functional interpretation is a young area with lots of challenges to overcome, but there are already some steps in this direction.

Some first steps

The synthesis of verified programs using proof interpretations like the functional interpretation is a young area with lots of challenges to overcome, but there are already some steps in this direction.

- Berger, U., Miyamoto, K., Schwichtenberg, H., and Seisenberger, M. (2011).

  Minlog - A tool for program extraction supporting algebras and coalgebras.
  In *Proceedings of CALCO 2011*, volume 6859 of *LNCS*, pages 393–399

Some first steps

The synthesis of verified programs using proof interpretations like the functional interpretation is a young area with lots of challenges to overcome, but there are already some steps in this direction.

- Berger, U., Miyamoto, K., Schwichtenberg, H., and Seisenberger, M. (2011).

  Minlog - A tool for program extraction supporting algebras and coalgebras.
  In *Proceedings of CALCO 2011*, volume 6859 of *LNCS*, pages 393–399

- Berger, U., Seisenberger, M., and Woods, G. (2014). Extracting imperative programs from proofs: In-place quicksort.
  In *Proceedings of TYPES 2013*, volume 26 of *LIPIcs*, pages 84–106

Some first steps

The synthesis of verified programs using proof interpretations like the functional interpretation is a young area with lots of challenges to overcome, but there are already some steps in this direction.

- Berger, U., Miyamoto, K., Schwichtenberg, H., and Seisenberger, M. (2011).

  Minlog - A tool for program extraction supporting algebras and coalgebras.
  In *Proceedings of CALCO 2011*, volume 6859 of *LNCS*, pages 393–399

- Berger, U., Seisenberger, M., and Woods, G. (2014). Extracting imperative programs from proofs: In-place quicksort.
  In *Proceedings of TYPES 2013*, volume 26 of *LIPIcs*, pages 84–106

- Berger, U., Miyamoto, K., Schwichtenberg, H., and Tsuiki, H. (2016). Logic for Gray-code computation.
  In *Concepts of Proof in Mathematics, Philosophy, and Computer Science*, pages 69–110. De Gruyter

Some first steps

The synthesis of verified programs using proof interpretations like the functional interpretation is a young area with lots of challenges to overcome, but there are already some steps in this direction.

- Berger, U., Miyamoto, K., Schwichtenberg, H., and Seisenberger, M. (2011).

  Minlog - A tool for program extraction supporting algebras and coalgebras.
  In *Proceedings of CALCO 2011*, volume 6859 of *LNCS*, pages 393–399

- Berger, U., Seisenberger, M., and Woods, G. (2014). Extracting imperative programs from proofs: In-place quicksort.
  In *Proceedings of TYPES 2013*, volume 26 of *LIPIcs*, pages 84–106

- Berger, U., Miyamoto, K., Schwichtenberg, H., and Tsuiki, H. (2016). Logic for Gray-code computation.
  In *Concepts of Proof in Mathematics, Philosophy, and Computer Science*, pages 69–110. De Gruyter

- Powell, T. A functional interpretation with state.
  To appear: Proceedings of Logic in Computer Science (LICS 2018)

Outline

References I

Avigad, J. (2009).
The metamathematics of ergodic theory.
Annals of Pure and Applied Logic, 157:64–76.

Baillot, P. and Das, A. (2016).
Free-cut elimination in linear logic and an application to a feasible arithmetic.
In *25th EACSL Annual Conference on Computer Science Logic, CSL 2016, August 29 - September 1, 2016, Marseille, France*, pages 40:1–40:18.

Bellantoni, S. and Cook, S. (1992).
A new recursion-theoretic characterization of the polytime functions.
Computational complexity, 2(2):97–110.

Berger, U., Miyamoto, K., Schwichtenberg, H., and Seisenberger, M. (2011).
Minlog - A tool for program extraction supporting algebras and coalgebras.
In *Proceedings of CALCO 2011*, volume 6859 of *LNCS*, pages 393–399.

Berger, U., Miyamoto, K., Schwichtenberg, H., and Tsuiki, H. (2016).
Logic for Gray-code computation.
In *Concepts of Proof in Mathematics, Philosophy, and Computer Science*, pages 69–110. De Gruyter.

References II

Berger, U., Seisenberger, M., and Woods, G. (2014).
Extracting imperative programs from proofs: In-place quicksort.
In *Proceedings of TYPES 2013*, volume 26 of *LIPIcs*, pages 84–106.

Burr, W. (1998).
Functionals in Set Theory and Arithmetic.
PhD thesis, Westfälische Wilhelms-Universität Münster.

Buss, S. R. (1986).
Bounded arithmetic.
PhD thesis.

Buss, S. R. (1991).
The witness function method and provably recursive functions of peano arithmetic.
In *Logic, Methodology and Philosophy of Science IX*, pages 29–68. North Holland.

Clote, P. and Kranakis, E. (2002).
Boolean Functions and Computation Models.

References III

Cook, S. and Nguyen, P. (2010).
Logical Foundations of Proof Complexity.
Cambridge University Press, New York, NY, USA, 1st edition.

Das, A. (2016).
From positive and intuitionistic bounded arithmetic to monotone proof complexity.
In *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science, LICS '16, New York, NY, USA, July 5-8, 2016*, pages 126–135.

Escardó, M. and Oliva, P. (2011).
Sequential games and optimal strategies.
Royal Society Proceedings A, 467:1519–1545.

Gaspar, J. and Kohlenbach, U. (2010).
On Tao's "finitary" infinite pigeonhole principle.
Journal of Symbolic Logic, 75(1):355–371.

Kohlenbach, K. and Oliva, P. (2003a).
Proof mining in the L_1-approximation.
Annals of Pure and Applied Logic, 121:1–38.

References IV

Kohlenbach, U.
Proof theoretic methods in nonlinear analysis.
To appear in: Proc. Int. Cong. of Math. - ICM 2018.

Kohlenbach, U. (1993a).
Effective moduli from ineffective uniqueness proofs. an unwinding of de la vallée poussin's proof for chebycheff approximation.
Annals of Pure and Applied Logic, 64:27–94.

Kohlenbach, U. (1993b).
New effective moduli of uniqueness and uniform a-priori estimates for constants of strong unicity by logical analysis of known proofs in best approximation theory.
Numer. Funct. Anal. Optim., 14:581–606.

Kohlenbach, U. (2008).
Applied Proof Theory - Proof Interpretations and their Use in Mathematics.
Springer Monographs in Mathematics. Springer.

Kohlenbach, U. and Oliva, P. (2003b).
A systematic way of analyzing proofs in mathematics.
Proceedings of the Steklov Institute of Mathematics, 242:136–164.

References V

Krajicek, J. (1995).
Bounded Arithmetic, Propositional Logic and Complexity Theory.
Encyclopedia of Mathematics and its Applications. Cambridge University Press.

Leivant, D. (1994).
Intrinsic theories and computational complexity.
In *Logical and Computational Complexity. Selected Papers. Logic and Computational Complexity, International Workshop LCC '94, Indianapolis, Indiana, USA, 13-16 October 1994*, pages 177–194.

Parsons, C. (1972).
On n-quantifier induction.
J. Symbolic Logic, 37(3):466–482.

Powell, T.
A functional interpretation with state.
To appear: Proceedings of Logic in Computer Science (LICS 2018).

Powell, T. (2013).
On Bar Recursive Interpretations of Analysis.
PhD thesis, Queen Mary University of London.

References VI

Powell, T. (2018).
Well quasi-orders and the functional interpretation.
To appear in: Schuster, P., Seisenberger, M. and Weiermann, A. editors, *Well Quasi-Orders in Computation, Logic, Language and Reasoning*, Trends in Logic, Springer.

Safarik, P. and Kohlenbach, U. (2010).
On the interpretation of the Bolzano-Weierstrass principle.
Mathematical Logic Quarterly, 56(5):508–532.

Sanders, S. (2018).
The Gandy-Hyland functional and a computational aspect of nonstandard analysis.
Computability, 7(1):7–43.

Spector, C. (1962).
Provably recursive functionals of analysis: a consistency proof of analysis by an extension of principles in current intuitionistic mathematics.
In Dekker, F. D. E., editor, *Recursive Function Theory: Proc. Symposia in Pure Mathematics*, volume 5, pages 1–27. American Mathematical Society, Providence, Rhode Island.

References VII

Tao, T. (2008a).
Soft analysis, hard analysis, and the finite convergence principle.
Essay, published as Ch. 1.3 of [Tao, 2008b], original version available online at `http://terrytao.wordpress.com/2007/05/23/soft-analysis-hard-analysis-and-the-finite-convergence-principle/`.

Tao, T. (2008b).
Structure and Randomness: Pages from Year 1 of a Mathematical Blog.
American Mathematical Society.

van den Berg, B., Briseid, E., and Safarik, P. (2012).
A functional interpretation for nonstandard arithmetic.
Annals of Pure and Applied Logic, 163(12):1962–1994.

van Heijenoort, J. (1972).
The collected papers of gerhard gentzen. m. e. szabo.
Philosophy of Science, 39(1):91–91.