Ideal objects and abstract machines

Thomas Powell

Technische Universität Darmstadt

Workshop on Computational Approaches to the Foundations of Mathematics

LMU Munich
13 April 2018

Proof $\mapsto$ Program

What *algorithm* is implemented by my extracted term?

What are algorithms and how do we characterise them?

Some ideas which influenced this talk:

* Gurevich's Abstract State Machines (ASMs)
* The Krivine machine (KAM) and classical realizability
* The treatment of ideal objects in constructive algebra
* U. Berger et al. - The formal extraction of sorting algorithms from proofs

Related work of my own:

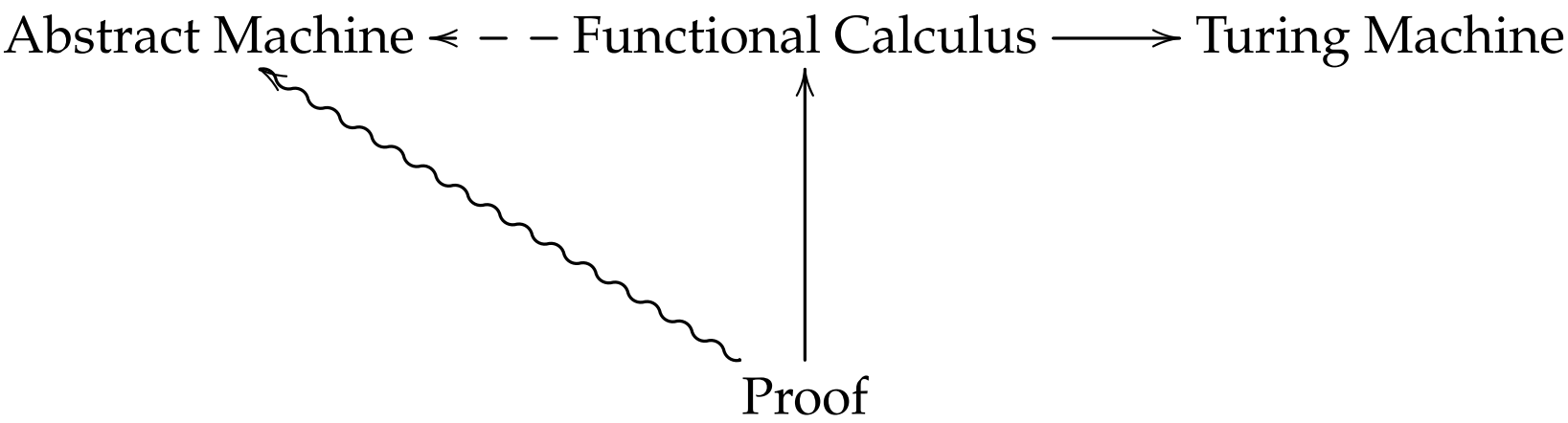
* *Gödel's functional interpretation and the concept of learning (P. LICS 16)*

Characterises the algorithmic behaviour of programs extracted from induction and weak comprehension principles, in terms of *higher-order learning procedures*.

* *A functional interpretation with state (P. LICS 18)*

Presents an imperative-style functional interpretation in which extracted programs interact with a global state, which captures what they learn from the 'mathematical environment'. Formalised using the state monad.

But everything still phrased in terms of extensions of System T...


Abstract Machine
Functional Calculus
Turing Machine
Proof

The rest of this talk consists of sketches from the last month or so...

Transition systems

At its simplest, a transition system consists of

- A set S of *states*
- A *transition relation* $\triangleright \subseteq S \times S$.

A computation is a sequence $s_0 \triangleright s_1 \triangleright \ldots \triangleright s_{n-1}$.

Transition systems

At its simplest, a transition system consists of

- A set S of *states*
- A *transition relation* $\triangleright \subseteq S \times S$.

A computation is a sequence $s_0 \triangleright s_1 \triangleright \ldots \triangleright s_{n-1}$.

$*$ Transition systems are very good at describing how programs work.

$*$ Our choice of transition system will depend on **what we are trying to describe**, and on **which level of abstraction**.

Kreisel's no-counterexample interpretation (n.c.i.)

For Π_3-formulas:

$$\underbrace{\forall a \exists x \forall y P(a,x,y)}_{(1)} \mapsto \underbrace{\forall a, f \exists x P(a,x,fx)}_{(2)}$$

(1) - There exist an ideal object x that works for all y;

(2) - For any f there exists an approximation x that works for fx.

Kreisel's no-counterexample interpretation (n.c.i.)

For Π_3-formulas:

$$\underbrace{\forall a \exists x \forall y P(a,x,y)}_{①} \mapsto \underbrace{\forall a, f \exists x P(a,x,fx)}_{②}$$

① - There exist an ideal object x that works for all y;

② - For any f there exists an approximation x that works for fx.

Our computational task: Construct a big enough approximation to the ideal object.

Extracted programs typically do this by trial and error (but this is often hidden!)...

So let's construct a transition system which captures this idea.

To keep in mind: $\forall a^{A}, f^{X \to Y} \exists x^{X}\ P(a, x, fx)$

So let's construct a transition system which captures this idea.

To keep in mind: $\forall a^{A}, f^{X \to Y} \exists x^{X}\ P(a, x, fx)$

A *state* encodes an object of type $S := C \times A \times X \times Y_{\square}$, where

- C is a set of commands which control the computation;

- A represents some background data (which is read-only);

- X is the object we are trying to compute, and the *input* for oracle queries;

- $Y_{\square} := Y \cup \{\square\}$ are values returned by oracle queries (possibly empty).

We write states as tuples $s := (c, a, x, y/\Box)$, where x is our 'current approximation'. Among our states we isolate:

- Initial states of the form $(c_0, a, x_0, \Box)$;

- Query states $S^?$ of the form $(c, a, x, \Box)$

- End states S^* of the form (c, a, x, y).

We write states as tuples $s := (c, a, x, y/\Box)$, where x is our 'current approximation'. Among our states we isolate:

- Initial states of the form $(c_0, a, x_0, \Box)$;

- Query states $S^?$ of the form $(c, a, x, \Box)$

- End states S^* of the form (c, a, x, y).

A *transition relation* $\triangleright$ is a partial function $\subseteq S \backslash (S^? \cup S^*) \times S$ (there are a few additional details)

States and transition relation together define what I will call an *abstract Approximation Machine* (AM)

$$\mathcal{M} = (C, A, X, Y, \triangleright)$$

An approximation machine $\mathcal{M}$ takes as input an arbitrary *oracle transition* $\blacktriangleright$, which is a partial function $\subseteq S^? \times S$.

A computation in $\mathcal{M}[\blacktriangleright]$ is a sequence of transitions of the form

- $(c, a, x, y) \triangleright (c', a, x', y/\square)$ computing approximation

- $(c, a, x, \square) \blacktriangleright (c, a, x, y)$ query status of current approximation

An approximation machine $\mathcal{M}$ takes as input an arbitrary *oracle transition* $\blacktriangleright$, which is a partial function $\subseteq S^? \times S$.

A computation in $\mathcal{M}[\blacktriangleright]$ is a sequence of transitions of the form

- $(c, a, x, y) \triangleright (c', a, x', y/\square)$ computing approximation

- $(c, a, x, \square) \blacktriangleright (c, a, x, y)$ query status of current approximation

We say that $\mathcal{M}$ computes $\forall a \exists x \forall y P(a, x, y)$ if for any $a, \blacktriangleright$ we have

$$(c_0, a, x_0, \square)(\triangleright \cup \blacktriangleright)^*(c, a, x, y) \in S^* \text{ such that } P(a, x, y).$$

Example: $\forall a^X \exists x^X \forall y^X (Q(a) \to Q(x) \wedge (y \prec x \to \neg Q(y)))$ for $\prec$ W.F.

EXAMPLE: $\forall a^X \exists x^X \forall y^X (Q(a) \to Q(x) \wedge (y \prec x \to \neg Q(y)))$ for $\prec$ W.F.

Define $S := \{\mathtt{I}, \mathtt{A}, \mathtt{E}\} \times X \times X \times X_\square$, with

- Initial states as $(\mathtt{I}, a, 0_X, \square)$;

- Query states as $(\mathtt{A}, a, x, \square)$;

- End states as $(\mathtt{E}, a, x, y)$.

Define $\triangleright$ by the following rules:

EXAMPLE: $\forall a^X \exists x^X \forall y^X (Q(a) \to Q(x) \wedge (y \prec x \to \neg Q(y)))$ for $\prec$ W.F.

Define $S := \{\mathtt{I}, \mathtt{A}, \mathtt{E}\} \times X \times X \times X_\square$, with

- Initial states as $(\mathtt{I}, a, 0_X, \square)$;

- Query states as $(\mathtt{A}, a, x, \square)$;

- End states as $(\mathtt{E}, a, x, y)$.

Define $\triangleright$ by the following rules:

$$(\mathtt{I}, a, 0_X, \square) \triangleright (\mathtt{A}, a, a, \square) \qquad \frac{y \prec x \wedge Q(y)}{(\mathtt{A}, a, x, y) \triangleright (\mathtt{A}, a, y, \square)} \qquad \frac{y \prec x \to \neg Q(y)}{(\mathtt{A}, a, x, y) \triangleright (\mathtt{E}, a, x, y)}$$

The resulting AM $\mathcal{M}$ computes $\forall a \exists x \forall y (Q(a) \to Q(x) \wedge (y \prec x \to \neg Q(y)))$.

To keep in mind: $\forall a^X \exists x^X \forall y^X (Q(a) \to Q(x) \wedge (y \prec x \to \neg Q(y)))$

Let $a, \blacktriangleright$ be arbitrary. We show that if $Q(x)$ holds then

$$(\mathtt{A}, a, x, \square)(\triangleright \cup \blacktriangleright)^*(\mathtt{E}, a, x', y) \text{ with } Q(x') \wedge (y \prec x' \to \neg Q(y))$$

To keep in mind: $\forall a^X \exists x^X \forall y^X (Q(a) \to Q(x) \wedge (y \prec x \to \neg Q(y)))$

Let $a, \blacktriangleright$ be arbitrary. We show that if $Q(x)$ holds then

$$(\mathtt{A}, a, x, \square)(\triangleright \cup \blacktriangleright)^*(\mathtt{E}, a, x', y) \text{ with } Q(x') \wedge (y \prec x' \to \neg Q(y))$$

First, we have $(\mathtt{A}, a, x, \square) \blacktriangleright (\mathtt{A}, a, x, y)$. If $y \prec x \to \neg Q(y)$ then $(\mathtt{A}, a, x, y) \triangleright (\mathtt{E}, a, x, y)$ and we're done.

To keep in mind: $\forall a^X \exists x^X \forall y^X (Q(a) \to Q(x) \wedge (y \prec x \to \neg Q(y)))$

Let $a, \blacktriangleright$ be arbitrary. We show that if $Q(x)$ holds then

$$(\mathtt{A}, a, x, \square)(\triangleright \cup \blacktriangleright)^*(\mathtt{E}, a, x', y) \text{ with } Q(x') \wedge (y \prec x' \to \neg Q(y))$$

First, we have $(\mathtt{A}, a, x, \square) \blacktriangleright (\mathtt{A}, a, x, y)$. If $y \prec x \to \neg Q(y)$ then $(\mathtt{A}, a, x, y) \triangleright (\mathtt{E}, a, x, y)$ and we're done.

Otherwise, $(\mathtt{A}, a, x, y) \triangleright (\mathtt{A}, a, y, \square)$ with $y \prec x$ and $Q(y)$, so use well-founded induction.

To keep in mind: $\forall a^X \exists x^X \forall y^X (Q(a) \to Q(x) \wedge (y \prec x \to \neg Q(y)))$

Let $a, \blacktriangleright$ be arbitrary. We show that if $Q(x)$ holds then

$$(\mathtt{A}, a, x, \square)(\triangleright \cup \blacktriangleright)^*(\mathtt{E}, a, x', y) \text{ with } Q(x') \wedge (y \prec x' \to \neg Q(y))$$

First, we have $(\mathtt{A}, a, x, \square) \blacktriangleright (\mathtt{A}, a, x, y)$. If $y \prec x \to \neg Q(y)$ then $(\mathtt{A}, a, x, y) \triangleright (\mathtt{E}, a, x, y)$ and we're done.

Otherwise, $(\mathtt{A}, a, x, y) \triangleright (\mathtt{A}, a, y, \square)$ with $y \prec x$ and $Q(y)$, so use well-founded induction.

Now, from the initial state we have $(\mathtt{I}, a, 0_X, \square) \triangleright (\mathtt{A}, a, a, \square)$, and assuming $Q(a)$ we're done.

The way in which our machine builds an approximation to a minimal element is completely transparent: The core of the computation consists of two moves:

- $(\mathtt{A}, a, x, \square) \blacktriangleright (\mathtt{A}, a, x, y)$ query the current approximation

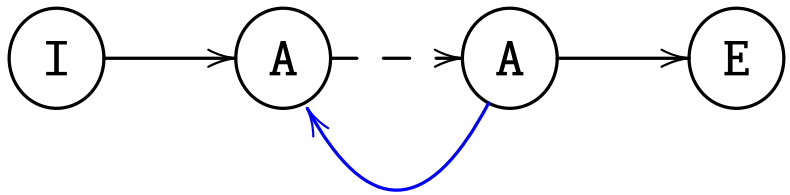
- $(\mathtt{A}, a, x, y) \triangleright (\mathtt{A}, a, y, \square)$ update the current approximation with a better one

The way in which our machine builds an approximation to a minimal element is completely transparent: The core of the computation consists of two moves:

- $(\mathtt{A}, a, x, \square) \blacktriangleright (\mathtt{A}, a, x, y)$ query the current approximation

- $(\mathtt{A}, a, x, y) \triangleright (\mathtt{A}, a, y, \square)$ update the current approximation with a better one

We can even draw a nice diagram of the control flow via our commands:



THEOREM. If a $\forall a \exists x \forall y P(a, x, y)$ is computed by an AM, then we can read off a functional Φ which realizes its n.c.i.

THEOREM. If a $\forall a \exists x \forall y P(a, x, y)$ is computed by an AM, then we can read off a functional Φ which realizes its n.c.i.

Proof. Given a function $f : X \to Y$, define $(c, a, x, \Box) \blacktriangleright_f (c, a, x, y)$ iff $y = fx$. Now let

$$\Phi a f := x \text{ where } (c_0, a, x_0, \Box)(\triangleright \cup \blacktriangleright_f)^*(c, a, x, y) \in S^*$$

THEOREM. If a $\forall a \exists x \forall y P(a, x, y)$ is computed by an AM, then we can read off a functional Φ which realizes its n.c.i.

Proof. Given a function $f : X \to Y$, define $(c, a, x, \Box) \blacktriangleright_f (c, a, x, y)$ iff $y = fx$. Now let

$$\Phi af := x \text{ where } (c_0, a, x_0, \Box)(\triangleright \cup \blacktriangleright_f)^*(c, a, x, y) \in S^*$$

But by definition we have $P(a, x, y)$, and moreover we must have $y = fx$ *(I cheated for this part: The definition of a AM is a tiny bit more complicated...).*

THEOREM. If a $\forall a \exists x \forall y P(a, x, y)$ is computed by an AM, then we can read off a functional Φ which realizes its n.c.i.

Proof. Given a function $f : X \to Y$, define $(c, a, x, \Box) \blacktriangleright_f (c, a, x, y)$ iff $y = fx$. Now let

$$\Phi af := x \text{ where } (c_0, a, x_0, \Box)(\triangleright \cup \blacktriangleright_f)^*(c, a, x, y) \in S^*$$

But by definition we have $P(a, x, y)$, and moreover we must have $y = fx$ *(I cheated for this part: The definition of a AM is a tiny bit more complicated...).*

Therefore $P(a, \Phi af, f(\Phi af))$. $\Box$

THEOREM. If a $\forall a \exists x \forall y P(a, x, y)$ is computed by an AM, then we can read off a functional Φ which realizes its n.c.i.

Proof. Given a function $f : X \to Y$, define $(c, a, x, \Box) \blacktriangleright_f (c, a, x, y)$ iff $y = fx$. Now let

$$\Phi af := x \text{ where } (c_0, a, x_0, \Box)(\triangleright \cup \blacktriangleright_f)^*(c, a, x, y) \in S^*$$

But by definition we have $P(a, x, y)$, and moreover we must have $y = fx$ *(I cheated for this part: The definition of a AM is a tiny bit more complicated...).*

Therefore $P(a, \Phi af, f(\Phi af))$. $\Box$

Remark 1. Having an approximation machine gives us additional information e.g. number of 'mind changes'.

THEOREM. If a $\forall a \exists x \forall y P(a, x, y)$ is computed by an AM, then we can read off a functional Φ which realizes its n.c.i.

Proof. Given a function $f : X \to Y$, define $(c, a, x, \Box) \blacktriangleright_f (c, a, x, y)$ iff $y = fx$. Now let

$$\Phi af := x \text{ where } (c_0, a, x_0, \Box)(\triangleright \cup \blacktriangleright_f)^*(c, a, x, y) \in S^*$$

But by definition we have $P(a, x, y)$, and moreover we must have $y = fx$ *(I cheated for this part: The definition of a AM is a tiny bit more complicated...).*

Therefore $P(a, \Phi af, f(\Phi af))$. $\Box$

Remark 1. Having an approximation machine gives us additional information e.g. number of 'mind changes'.

Remark 2. Some kind of converse also holds given continuity assumptions, but that's not the point of the approximation machine!

So what is the point of a thing like the approximation machine?

One possibility - A fully automatic procedure:

$$\mathcal{T} \vdash A \Rightarrow \exists \mathcal{M} \text{ such that } \mathcal{M} \text{ computes } A$$

where $\mathcal{M}$ can be extracted from the proof of A.

One possibility - A fully automatic procedure:

$$\mathcal{T} \vdash A \Rightarrow \exists \mathcal{M} \text{ such that } \mathcal{M} \text{ computes } A$$

where $\mathcal{M}$ can be extracted from the proof of A.

On the other hand, we could just use it a way of describing some programs which we already understand.

$$\Phi \text{ realizes the n.c.i. of } A, \text{ and is simulated by the AM } \mathcal{M}$$

One possibility - A fully automatic procedure:

$$\mathcal{T} \vdash A \Rightarrow \exists \mathcal{M} \text{ such that } \mathcal{M} \text{ computes } A$$

where $\mathcal{M}$ can be extracted from the proof of A.

On the other hand, we could just use it a way of describing some programs which we already understand.

$$\Phi \text{ realizes the n.c.i. of } A, \text{ and is simulated by the AM } \mathcal{M}$$

Ideally, though, we would have a combination: A set of operations which take simple machines $\mathcal{M}$ and produce complex ones.

$$\text{if } \mathcal{M} \text{ computes } A \text{ and } \frac{A}{B} \text{ then } \mathcal{M}^\star \text{ computes } B$$

Rule of dependent choice

$$\frac{\forall a^{X^*} \exists x^{X} \forall y^{Y} P(a,x,y)}{\exists f^{\mathbb{N}\to X} \forall n^{\mathbb{N}}, y P(\bar{f}n, fn, y)} \sim \frac{\mathcal{M}}{\mathcal{M}^{\star}}$$

Rule of dependent choice

$$\frac{\forall a^{X^*} \exists x^X \forall y^Y P(a,x,y)}{\exists f^{\mathbb{N}\to X} \forall n^{\mathbb{N}}, y P(\bar{f}n, fn, y)} \sim \frac{\mathcal{M}}{\mathcal{M}^\star}$$

Note that the n.c.i. of the denominator is

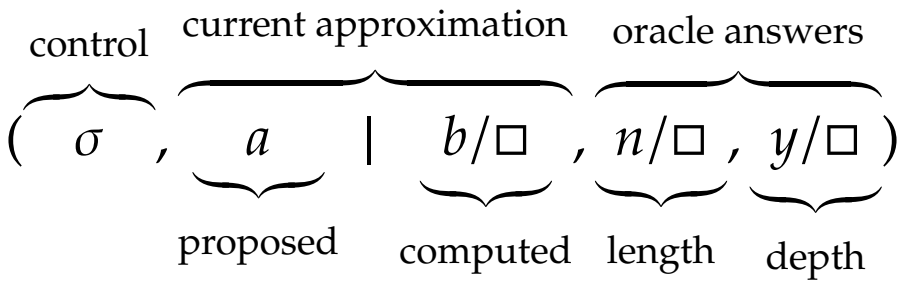
$$\forall \omega, \phi \exists f \; P(\bar{f}(\omega f), f(\omega f), \phi f)$$

i.e. there exists an approximate choice sequence f which works at point $n := \omega f$ and for $y := \phi f$.

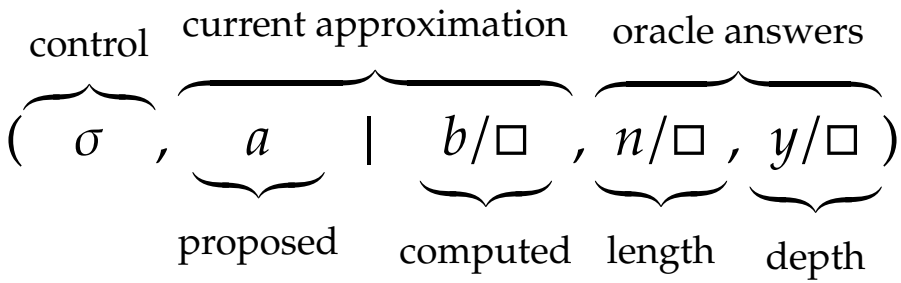
- ω characterises the 'length' of the approximation;

- ϕ characterises the 'depth' of the approximation.

Let $\mathcal{M} = (C, X^*, X, Y, \triangleright)$. We define $\mathcal{M}^\star$ to have states of the form



Let $\mathcal{M} = (C, X^*, X, Y, \triangleright)$. We define $\mathcal{M}^\star$ to have states of the form



- the control represents a sequence $\sigma := [c_0, \ldots, c_{n-1}]$ of continuations;

Let $\mathcal{M} = (C, X^*, X, Y, \triangleright)$. We define $\mathcal{M}^\star$ to have states of the form

$$(\overbrace{\sigma}^{\text{control}}, \overbrace{\underbrace{a}_{\text{proposed}} \mid \underbrace{b/\square}_{\text{computed}}}^{\text{current approximation}}, \overbrace{\underbrace{n/\square}_{\text{length}}, \underbrace{y/\square}_{\text{depth}}}^{\text{oracle answers}})$$

- the control represents a sequence $\sigma := [c_0, \ldots, c_{n-1}]$ of continuations;

- the main object being computed is a pair of finite sequence $a \mid b$ which represent the choice sequence $a :: b :: 0, 0, \ldots$. We view b as the current 'completion' of a.

Let $\mathcal{M} = (C, X^*, X, Y, \triangleright)$. We define $\mathcal{M}^\star$ to have states of the form

$$(\overbrace{\sigma}^{\text{control}}, \overbrace{\underbrace{a}_{\text{proposed}} \mid \underbrace{b/\square}_{\text{computed}}}^{\text{current approximation}}, \overbrace{\underbrace{n/\square}_{\text{length}}, \underbrace{y/\square}_{\text{depth}}}^{\text{oracle answers}})$$

- the control represents a sequence $\sigma := [c_0, \ldots, c_{n-1}]$ of continuations;

- the main object being computed is a pair of finite sequence $a \mid b$ which represent the choice sequence $a :: b :: 0, 0, \ldots$. We view b as the current 'completion' of a.

- We now have two oracles, which take the approximation $a \mid b$ and return the desired length and depth respectively, which eventually need to be satisfied by our approximation...

In state $(\sigma :: c, a :: x \mid b/\square, n/\square, y/\square)$, x is the element currently being computed.

In state $(\sigma :: c, a :: x \mid b/\square, n/\square, y/\square)$, x is the element currently being computed.

If $(c, a, x, \square) \in S^?$ then

In state $(\sigma :: c, a :: x \mid b/\square, n/\square, y/\square)$, x is the element currently being computed.

If $(c, a, x, \square) \in S^?$ then

- $(\sigma :: c, a :: x \mid \square, \square, \square) \blacktriangleright_l (\sigma :: c, a :: x \mid \square, n, \square)$ check length

In state $(\sigma :: c, a :: x \mid b/\square, n/\square, y/\square)$, x is the element currently being computed.

If $(c, a, x, \square) \in S^?$ then

- $(\sigma :: c, a :: x \mid \square, \square, \square) \blacktriangleright_l (\sigma :: c, a :: x \mid \square, n, \square)$ check length

- If $n < |a|$ then $(\sigma :: c, a :: x \mid \square, n, \square) \rhd_\star (\sigma :: c, a :: x \mid [], n, \square)$ length good

In state $(\sigma :: c, a :: x \mid b/\square, n/\square, y/\square)$, x is the element currently being computed.

If $(c, a, x, \square) \in S^?$ then

- $(\sigma :: c, a :: x \mid \square, \square, \square) \blacktriangleright_l (\sigma :: c, a :: x \mid \square, n, \square)$ check length

- If $n < |a|$ then $(\sigma :: c, a :: x \mid \square, n, \square) \rhd_\star (\sigma :: c, a :: x \mid [], n, \square)$ length good

- If $n \geq |a|$ then $(\sigma :: c, a :: x \mid \square, n, \square) \rhd_\star (\sigma :: c :: c_0, a :: x :: x_0 \mid \square, \square, \square)$
length bad

In state $(\sigma :: c, a :: x \mid b/\square, n/\square, y/\square)$, x is the element currently being computed.

If $(c, a, x, \square) \in S^?$ then

- $(\sigma :: c, a :: x \mid \square, \square, \square) \blacktriangleright_l (\sigma :: c, a :: x \mid \square, n, \square)$ check length

- If $n < |a|$ then $(\sigma :: c, a :: x \mid \square, n, \square) \triangleright_\star (\sigma :: c, a :: x \mid [], n, \square)$ length good

- If $n \geq |a|$ then $(\sigma :: c, a :: x \mid \square, n, \square) \triangleright_\star (\sigma :: c :: c_0, a :: x :: x_0 \mid \square, \square, \square)$ length bad

- $(\sigma :: c, a :: x \mid [], n, \square) \blacktriangleright_d (\sigma :: c, a :: x \mid [], n, y)$ check depth

In state $(\sigma :: c, a :: x \mid b/\square, n/\square, y/\square)$, x is the element currently being computed.

If $(c, a, x, \square) \in S^?$ then

- $(\sigma :: c, a :: x \mid \square, \square, \square) \blacktriangleright_l (\sigma :: c, a :: x \mid \square, n, \square)$ check length

- If $n < |a|$ then $(\sigma :: c, a :: x \mid \square, n, \square) \rhd_\star (\sigma :: c, a :: x \mid [], n, \square)$ length good

- If $n \geq |a|$ then $(\sigma :: c, a :: x \mid \square, n, \square) \rhd_\star (\sigma :: c :: c_0, a :: x :: x_0 \mid \square, \square, \square)$ length bad

- $(\sigma :: c, a :: x \mid [], n, \square) \blacktriangleright_d (\sigma :: c, a :: x \mid [], n, y)$ check depth

If $(c, a, x, y) \rhd (c', a, x', y/\square)$ then

In state $(\sigma :: c, a :: x \mid b/\square, n/\square, y/\square)$, x is the element currently being computed.

If $(c, a, x, \square) \in S^?$ then

- $(\sigma :: c, a :: x \mid \square, \square, \square) \blacktriangleright_l (\sigma :: c, a :: x \mid \square, n, \square)$ check length

- If $n < |a|$ then $(\sigma :: c, a :: x \mid \square, n, \square) \rhd_\star (\sigma :: c, a :: x \mid [], n, \square)$ length good

- If $n \geq |a|$ then $(\sigma :: c, a :: x \mid \square, n, \square) \rhd_\star (\sigma :: c :: c_0, a :: x :: x_0 \mid \square, \square, \square)$ length bad

- $(\sigma :: c, a :: x \mid [], n, \square) \blacktriangleright_d (\sigma :: c, a :: x \mid [], n, y)$ check depth

If $(c, a, x, y) \rhd (c', a, x', y/\square)$ then

- $(\sigma :: c, a :: x \mid b, n, y) \rhd_\star (\sigma :: c', a :: x' \mid b/\square, n/\square, y/\square)$ compute element

In state $(\sigma :: c, a :: x \mid b/\square, n/\square, y/\square)$, x is the element currently being computed.

If $(c, a, x, \square) \in S^?$ then

- $(\sigma :: c, a :: x \mid \square, \square, \square) \blacktriangleright_l (\sigma :: c, a :: x \mid \square, n, \square)$ check length

- If $n < |a|$ then $(\sigma :: c, a :: x \mid \square, n, \square) \rhd_\star (\sigma :: c, a :: x \mid [], n, \square)$ length good

- If $n \geq |a|$ then $(\sigma :: c, a :: x \mid \square, n, \square) \rhd_\star (\sigma :: c :: c_0, a :: x :: x_0 \mid \square, \square, \square)$ length bad

- $(\sigma :: c, a :: x \mid [], n, \square) \blacktriangleright_d (\sigma :: c, a :: x \mid [], n, y)$ check depth

If $(c, a, x, y) \rhd (c', a, x', y/\square)$ then

- $(\sigma :: c, a :: x \mid b, n, y) \rhd_\star (\sigma :: c', a :: x' \mid b/\square, n/\square, y/\square)$ compute element

If $(c, a, x, y) \in S^*$ then

In state $(\sigma :: c, a :: x \mid b/\square, n/\square, y/\square)$, x is the element currently being computed.

If $(c, a, x, \square) \in S^?$ then

- $(\sigma :: c, a :: x \mid \square, \square, \square) \blacktriangleright_l (\sigma :: c, a :: x \mid \square, n, \square)$ check length

- If $n < |a|$ then $(\sigma :: c, a :: x \mid \square, n, \square) \rhd_\star (\sigma :: c, a :: x \mid [], n, \square)$ length good

- If $n \geq |a|$ then $(\sigma :: c, a :: x \mid \square, n, \square) \rhd_\star (\sigma :: c :: c_0, a :: x :: x_0 \mid \square, \square, \square)$ length bad

- $(\sigma :: c, a :: x \mid [], n, \square) \blacktriangleright_d (\sigma :: c, a :: x \mid [], n, y)$ check depth

If $(c, a, x, y) \rhd (c', a, x', y/\square)$ then

- $(\sigma :: c, a :: x \mid b, n, y) \rhd_\star (\sigma :: c', a :: x' \mid b/\square, n/\square, y/\square)$ compute element

If $(c, a, x, y) \in S^*$ then

- $(\sigma :: c, a :: x \mid b, n, y) \rhd_\star (\sigma, a \mid x :: b, n, y)$ element computed

THEOREM. Suppose that $\mathcal{M}$ computes $\forall a^{X^*} \exists x^X \forall y^Y P(a,x,y)$ i.e. for any $a, \blacktriangleright$

$$(c_0, a, x_0, \square)(\triangleright \cup \blacktriangleright)^*(c, a, x, y) \in S^* \text{ with } P(a,x,y)$$

Then for any (continuous!) $\blacktriangleright_l, \blacktriangleright_d$, we have

$$(\sigma, a \mid \square, \square, \square)(\triangleright_\star \cup \blacktriangleright_l \cup \blacktriangleright_d)^*(\sigma, a \mid b, n, y)$$

with (i) $n < |a| + |b|$ and (ii) $\forall i < |b|\ P(a :: \bar{b}i, bi, y)$.

Theorem. Suppose that $\mathcal{M}$ computes $\forall a^{X^*} \exists x^X \forall y^Y P(a,x,y)$ i.e. for any $a, \blacktriangleright$

$$(c_0, a, x_0, \square)(\triangleright \cup \blacktriangleright)^*(c, a, x, y) \in S^* \text{ with } P(a,x,y)$$

Then for any (continuous!) $\blacktriangleright_l, \blacktriangleright_d$, we have

$$(\sigma, a \mid \square, \square, \square)(\triangleright_\star \cup \blacktriangleright_l \cup \blacktriangleright_d)^*(\sigma, a \mid b, n, y)$$

with (i) $n < |a| + |b|$ and (ii) $\forall i < |b|\ P(a :: \bar{b}i, bi, y)$.

Corollary. We have

$$([], [] \mid \square, \square, \square)(\triangleright_* \cup \blacktriangleright_l \cup \blacktriangleright_d)^*([], [] \mid b, n, y)$$

with (i) $n < |b|$ and $\forall i < |b|\ P(\bar{b}i, bi, y)$.

Theorem. Suppose that $\mathcal{M}$ computes $\forall a^{X^*} \exists x^X \forall y^Y P(a,x,y)$ i.e. for any $a, \blacktriangleright$

$$(c_0, a, x_0, \square)(\triangleright \cup \blacktriangleright)^*(c, a, x, y) \in S^* \text{ with } P(a,x,y)$$

Then for any (continuous!) $\blacktriangleright_l, \blacktriangleright_d$, we have

$$(\sigma, a \mid \square, \square, \square)(\triangleright_\star \cup \blacktriangleright_l \cup \blacktriangleright_d)^*(\sigma, a \mid b, n, y)$$

with (i) $n < |a| + |b|$ and (ii) $\forall i < |b|\ P(a :: \bar{b}i, bi, y)$.

Corollary. We have

$$([], [] \mid \square, \square, \square)(\triangleright_* \cup \blacktriangleright_l \cup \blacktriangleright_d)^*([], [] \mid b, n, y)$$

with (i) $n < |b|$ and $\forall i < |b|\ P(\bar{b}i, bi, y)$.

Define $\blacktriangleright_l, \blacktriangleright_d$ in terms of $\lambda s.\omega\hat{s}, \lambda s.\phi\hat{s}$ respectively, where $\hat{s} := s :: 0, 0, 0, \ldots$, and let $f := \hat{b}$.

Then $n, y = \omega f, \phi f$ and $P(\bar{f}(\omega f), f(\omega f), \phi f)$ holds.

Now for a real example...

Classic theorem. Let R be a commutative ring with $0 \neq 1$. Suppose that r lies in the intersection of all prime ideals of R. Then r is nilpotent ($\exists e > 0(r^e = 0)$).

Classic theorem. Let R be a commutative ring with $0 \neq 1$. Suppose that r lies in the intersection of all prime ideals of R. Then r is nilpotent ($\exists e > 0(r^e = 0)$).

Proof (sketch). Suppose that r is not nilpotent. Define

$$\Sigma := \{I \subset R \mid I \text{ is an ideal satisfying } \forall e > 0(r^e \notin I)\}.$$

Then $\{0\} \in \Sigma$ (by our assumption), and Σ is chain-complete w.r.t. inclusion, so by Zorn's lemma it has a maximal element M.

Classic theorem. Let R be a commutative ring with $0 \neq 1$. Suppose that r lies in the intersection of all prime ideals of R. Then r is nilpotent ($\exists e > 0(r^e = 0)$).

Proof (sketch). Suppose that r is not nilpotent. Define

$$\Sigma := \{I \subset R \mid I \text{ is an ideal satisfying } \forall e > 0(r^e \notin I)\}.$$

Then $\{0\} \in \Sigma$ (by our assumption), and Σ is chain-complete w.r.t. inclusion, so by Zorn's lemma it has a maximal element M.

We show that M is prime: If $m, n \notin M$ then $M + (m)$ and $M + (n)$ are proper extensions of M, so by maximality there exist $e_1, e_2 > 0$ such that $r^{e_1} \in M + (m)$ and $r^{e_2} \in M + (n)$. Therefore

$$r^{e_1+e_2} \in M + (mn)$$

and so $M + (mn) \notin \Sigma$, which means that $mn \notin M$.

Classic theorem. Let R be a commutative ring with $0 \neq 1$. Suppose that r lies in the intersection of all prime ideals of R. Then r is nilpotent ($\exists e > 0(r^e = 0)$).

Proof (sketch). Suppose that r is not nilpotent. Define

$$\Sigma := \{I \subset R \mid I \text{ is an ideal satisfying } \forall e > 0(r^e \notin I)\}.$$

Then $\{0\} \in \Sigma$ (by our assumption), and Σ is chain-complete w.r.t. inclusion, so by Zorn's lemma it has a maximal element M.

We show that M is prime: If $m, n \notin M$ then $M + (m)$ and $M + (n)$ are proper extensions of M, so by maximality there exist $e_1, e_2 > 0$ such that $r^{e_1} \in M + (m)$ and $r^{e_2} \in M + (n)$. Therefore

$$r^{e_1+e_2} \in M + (mn)$$

and so $M + (mn) \notin \Sigma$, which means that $mn \notin M$.

Since $r^1 \notin M$, r cannot lie in the intersection of all prime ideals. □

For countable commutative rings $R := \{r_n \ : \ n \in \mathbb{N}\}$ (with w.l.o.g. $r_0 = 0$), this proof can be formalised in ACA_0 via a standard trick in reverse math (N.B. I didn't find out whether the theorem itself is strictly weaker...)

There is a corresponding Approximation Machine:

$$(\sigma, \underbrace{a \mid b}_{\text{maximal } M \in \Sigma}, \underbrace{n, \langle [y_1, \ldots, y_k, y], e \rangle}_{m_1 y_1 + \ldots + m_k y_k + r_n y = r^e}) \text{ with } a(i) = \langle \underbrace{\chi(i)}_{\mathbb{B}}, \underbrace{\vec{y}(i)}_{R^*}, \underbrace{e(i)}_{\mathbb{N}_{>0}} \rangle$$

For countable commutative rings $R := \{r_n \ : \ n \in \mathbb{N}\}$ (with w.l.o.g. $r_0 = 0$), this proof can be formalised in ACA_0 via a standard trick in reverse math (N.B. I didn't find out whether the theorem itself is strictly weaker...)

There is a corresponding Approximation Machine:

$$(\sigma, \underbrace{a \mid b}_{\text{maximal } M \in \Sigma}, \underbrace{n, \langle [y_1, \ldots, y_k, y], e \rangle}_{m_1 y_1 + \ldots + m_k y_k + r_n y = r^e}) \text{ with } a(i) = \langle \underbrace{\chi(i)}_{\mathbb{B}}, \underbrace{\vec{y}(i)}_{R^*}, \underbrace{e(i)}_{\mathbb{N}_{>0}} \rangle$$

- Either $\chi(i) = 1$ and $r_i \in M$, or $\chi(i) = 0$ and $r_i \notin M$, in which case $\langle \vec{y}(i), e(i) \rangle$ act as *evidence* for the exclusion of r_i i.e.

$$(\bar{m}i \cup \{r_i\}) \cdot \vec{y}(i) = r^{e(i)}$$

For countable commutative rings $R := \{r_n \ : \ n \in \mathbb{N}\}$ (with w.l.o.g. $r_0 = 0$), this proof can be formalised in ACA_0 via a standard trick in reverse math (N.B. I didn't find out whether the theorem itself is strictly weaker...)

There is a corresponding Approximation Machine:

$$(\sigma, \underbrace{a \mid b}_{\text{maximal } M \in \Sigma}, \underbrace{n, \langle [y_1, \ldots, y_k, y], e \rangle}_{m_1 y_1 + \ldots + m_k y_k + r_n y = r^e}) \text{ with } a(i) = \langle \underbrace{\chi(i)}_{\mathbb{B}}, \underbrace{\vec{y}(i)}_{R^*}, \underbrace{e(i)}_{\mathbb{N}_{>0}} \rangle$$

- Either $\chi(i) = 1$ and $r_i \in M$, or $\chi(i) = 0$ and $r_i \notin M$, in which case $\langle \vec{y}(i), e(i) \rangle$ act as *evidence* for the exclusion of r_i i.e.

$$(\bar{m}i \cup \{r_i\}) \cdot \vec{y}(i) = r^{e(i)}$$

- Oracle queries $\blacktriangleright_l, \blacktriangleright_d$ provide evidence that for a given M encoded by $a \mid b$, we have

$$r \in M \vee \ M \text{ not a prime ideal}$$

which is converted into evidence for excluding r_n for some $n \in \mathbb{N}$.

Theorem. We have

$$([], \underbrace{[] \mid \square}_{M:=R}, \square, \square)(\triangleright_\star \cup \blacktriangleright_l \cup \blacktriangleright_d)^*([], \underbrace{[] \mid b}_{M \subset R}, n, \langle [y_1, \ldots, y_k, y], e \rangle)$$

such that $r_0 = 0 \notin M$ i.e. $b(0) = \langle 0, [y_0], e(0) \rangle$ with $e(0) > 0$ such that

$$r_0 y_0 = r^{e(0)} \text{ i.e. } r^{e(0)} = 0.$$

THEOREM. We have

$$([], \underbrace{[] \mid \square}_{M:=R}, \square, \square)(\rhd_\star \cup \blacktriangleright_l \cup \blacktriangleright_d)^*([], \underbrace{[] \mid b}_{M \subset R}, n, \langle [y_1, \ldots, y_k, y], e \rangle)$$

such that $r_0 = 0 \notin M$ i.e. $b(0) = \langle 0, [y_0], e(0) \rangle$ with $e(0) > 0$ such that

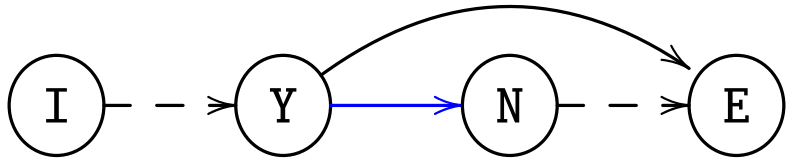
$$r_0 y_0 = r^{e(0)} \text{ i.e. } r^{e(0)} = 0.$$

Our AM forms the core of a computational version of the classical proof:

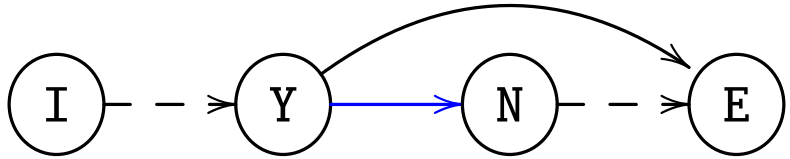
- **Classical.** There exists a maximal element $M \in \Sigma$, hence $r^e = 0$ for some $e > 0$ by contradiction.

- **Computational.** There exists an AM with parameters $\blacktriangleright_l, \blacktriangleright_d$ which builds an approximation to M by starting with $M := R$ and gradually excluding elements. Eventually 0 is excluded, hence we have found via the $\blacktriangleright_l, \blacktriangleright_r$ some $e > 0$ with $r^e = 0$.

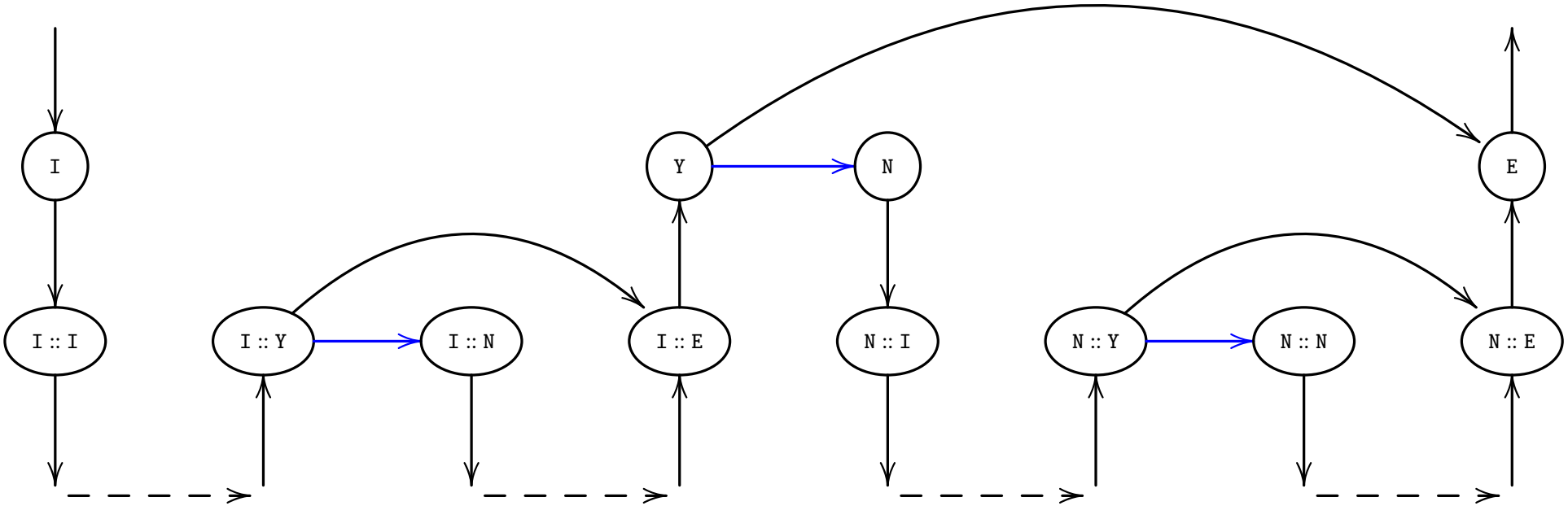
Input machine $\mathcal{M}$ for single elements:



Input machine $\mathcal{M}$ for single elements:



Output machine $\mathcal{M}^\star$ for whole set:



Open questions

Essentially everything! But in particular:

- What about machines that capture other patterns we see in extracted programs?

- What about the full Dialectica interpretations i.e. oracle machines in all finite types?

- Can we give a formal geometric characterisation of what's going on via some kind of graphs?

Thank You!