Automated Equational Reasoning in Nondeterministic λ-Calculi Modulo Theories $\mathcal{H}^*$

Fritz Obermeyer

Department of Mathematics
Carnegie Mellon University

2009:03:22

Motivation

to semi-automatically build

a complete knowledge base

of mathematical facts

Motivation

to semi-automatically build

a complete knowledge base

of mathematical facts

Motivation

to semi-automatically build

a complete knowledge base

of mathematical facts

Motivation

to semi-automatically build

a complete knowledge base

of mathematical facts

Motivation

to semi-automatically build

a complete knowledge base

of mathematical facts

Motivation

to semi-automatically build

a complete knowledge base

of mathematical facts

What do we want of a foundation?

What do we want of a foundation?

Simple syntax

What do we want of a foundation?

Simple syntax *combinators*

What do we want of a foundation?

Simple syntax *combinators*

Full extensionality

What do we want of a foundation?

Simple syntax *combinators*

Full extensionality $\mathcal{H}^*$

What is $\mathcal{H}^*$?

Add probing term $\top\ x = \top$.

What is $\mathcal{H}^*$?

Add probing term $\top$ $\mathtt{x} = \top$.

$$\frac{\mathtt{x} = \top}{\mathtt{x}\ \ \mathtt{conv}} \qquad \frac{\mathtt{x}\ \ \top\ \ \mathtt{conv}}{\mathtt{x}\ \ \mathtt{conv}}$$

What is $\mathcal{H}^*$?

Add probing term $\top$ x $= \top$.

$$\frac{x = \top}{x \ \text{conv}} \qquad \frac{x \ \top \ \text{conv}}{x \ \text{conv}}$$

λx.x

What is $\mathcal{H}^*$?

Add probing term $\top\ \mathsf{x} = \top$.

$$\frac{\mathsf{x} = \top}{\mathsf{x}\ \mathsf{conv}} \qquad \frac{\mathsf{x}\ \top\ \mathsf{conv}}{\mathsf{x}\ \mathsf{conv}}$$

$$\lambda \mathsf{x}.\mathsf{x} \quad \mapsto \quad (\lambda \mathsf{x}.\mathsf{x})\top$$

What is $\mathcal{H}^*$?

Add probing term $\top$ $x = \top$.

$$\frac{x = \top}{x \ \text{conv}} \qquad \frac{x \ \top \ \text{conv}}{x \ \text{conv}}$$

$$\lambda x.x \quad \mapsto \quad (\lambda x.x)\top = \top$$

What is $\mathcal{H}^*$?

Add probing term $\top\ \ x = \top$.

$$\frac{x = \top}{x\ \text{conv}} \qquad \frac{x\ \ \top\ \ \text{conv}}{x\ \text{conv}}$$

$$\lambda x.x \quad \mapsto \quad (\lambda x.x)\top = \top \qquad \text{converges}$$

What is $\mathcal{H}^*$?

Add probing term $\top$ x $= \top$.

$$\frac{x = \top}{x \ \text{conv}} \qquad \frac{x \ \top \ \text{conv}}{x \ \text{conv}}$$

$\lambda x.x \ \mapsto \ (\lambda x.x)\top = \top$ converges

$(\lambda x.xx)(\lambda x.xx)$

What is $\mathcal{H}^*$?

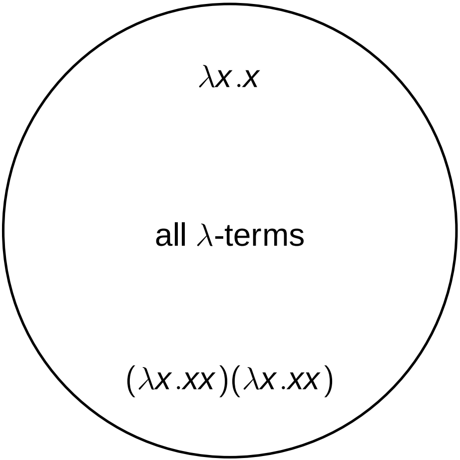
Add probing term $\top\ x = \top$.

$$\frac{x = \top}{x\ \text{conv}} \qquad \frac{x\ \top\ \text{conv}}{x\ \text{conv}}$$

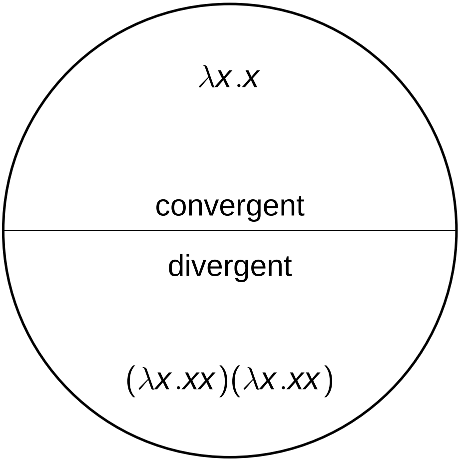
$\lambda x.x \ \mapsto\ (\lambda x.x)\top = \top$ converges

$(\lambda x.xx)(\lambda x.xx)$ diverges

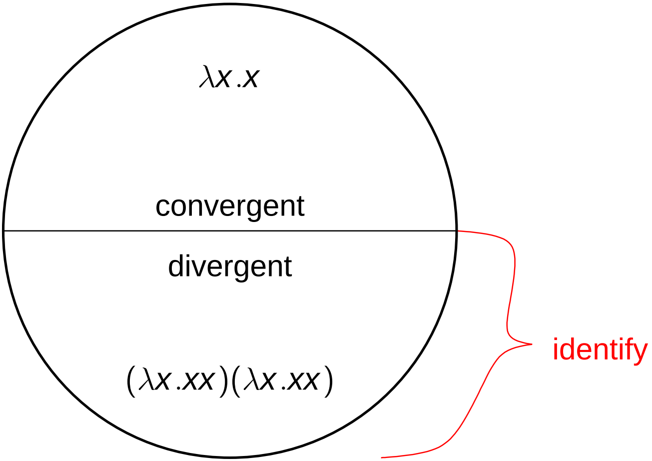
What is $\mathcal{H}^*$?



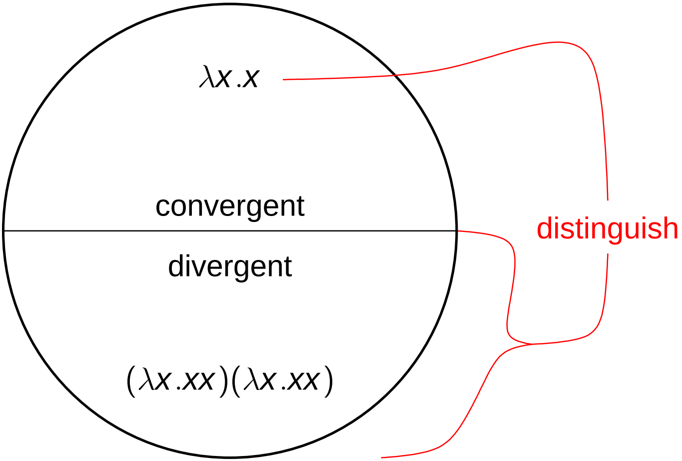
What is $\mathcal{H}^*$?



What is $\mathcal{H}^*$?



What is $\mathcal{H}^*$?



What do we want out of a foundation?

Simple syntax *combinators*

Full extensionality $\mathcal{H}^*$

What do we want out of a foundation?

Simple syntax *combinators*

Full extensionality $\mathcal{H}^*$

A universe for all math

What do we want out of a foundation?

Simple syntax	*combinators*
Full extensionality	$\mathcal{H}^*$
A universe for all math	*varies...*

Outline

$\mathcal{H}^*$ for λ-join-calculus

Generalize to order

$$\frac{\forall \mathbf{C}[\ ].\ \mathbf{C}[x]\ \text{conv} \iff \mathbf{C}[y]\ \text{conv}}{\mathcal{H}^* \ \vdash\ x = y}$$

$\mathcal{H}^*$ for λ-join-calculus

Generalize to order

$$\frac{\forall \mathbf{C}[\ ].\ \ \mathbf{C}[x]\ \text{conv} \quad \Longrightarrow \quad \mathbf{C}[y]\ \text{conv}}{\mathcal{H}^* \ \vdash \ x \sqsubseteq y}$$

$\mathcal{H}^*$ for λ-join-calculus

Generalize to order, traces $\mathbf{C}[x] = x\ M_1\ \ldots\ M_n$

$$\frac{\forall \mathbf{C}[\ ].\ \mathbf{C}[x]\ \text{conv} \implies \mathbf{C}[y]\ \text{conv}}{\mathcal{H}^* \vdash x \sqsubseteq y}$$

$\mathcal{H}^*$ for λ-join-calculus

Generalize to order, traces $\mathbf{C}[x] = x\ M_1\ \ldots\ M_n$

$$\frac{\forall \underline{M}.\ x\ \underline{M}\ \text{conv} \implies y\ \underline{M}\ \text{conv}}{\mathcal{H}^* \vdash x \sqsubseteq y}$$

$\mathcal{H}^*$ for λ-join-calculus

Generalize to order, traces $\mathbf{C}[x] = x\ M_1\ \ldots\ M_n$

$$\frac{\forall \underline{M}.\ x\ \underline{M}\ \text{conv} \implies y\ \underline{M}\ \text{conv}}{\mathcal{H}^* \vdash x \sqsubseteq y}$$

Add binary join: $(x \mid y)z = x\ z \mid y\ z$

$\mathcal{H}^*$ for λ-join-calculus

Generalize to order, traces $\mathbf{C}[x] = x\ M_1\ \ldots\ M_n$

$$\frac{\forall \underline{M}.\ x\ \underline{M}\ \text{conv} \implies y\ \underline{M}\ \text{conv}}{\mathcal{H}^* \vdash x \sqsubseteq y}$$

Add binary join: $(x \mid y)z = x\ z \mid y\ z$

$x \mid y$ conv whenever x conv or y conv

Types-as-closures

closures $\mathbf{I} \sqsubseteq a = a \circ a,$

Types-as-closures

$$\text{closures} \qquad \mathbf{I} \sqsubseteq \mathsf{a} = \mathsf{a} \circ \mathsf{a},$$
$$\mathsf{x} : \mathsf{a} \iff \mathsf{a}\ \mathsf{x} = \mathsf{x}$$

Types-as-closures

closures $\mathbf{I} \sqsubseteq \mathsf{a} = \mathsf{a} \circ \mathsf{a}$,

$\mathsf{x} : \mathsf{a} \iff \mathsf{a}\ \mathsf{x} = \mathsf{x}$

universe: $\mathbf{V} = \lambda \mathsf{a}.\ \mathbf{I} \mid \mathsf{a} \mid \mathsf{a} \circ \mathsf{a} \mid \ldots,$

Types-as-closures

$$\text{closures} \qquad \mathbf{I} \sqsubseteq a = a \circ a,$$
$$x : a \iff a\ x = x$$

$$\text{universe:} \qquad \mathbf{V} = \lambda a.\ \mathbf{I} \mid a \mid a \circ a \mid \ldots,$$
$$\text{exponentials:} \qquad a \to b = \lambda f. b \circ f \circ a,$$

Types-as-closures

closures $\mathbf{I} \sqsubseteq a = a \circ a,$

$x : a \iff a \; x = x$

universe: $\mathbf{V} = \lambda a. \; \mathbf{I} \mid a \mid a \circ a \mid \ldots,$

exponentials: $a \to b = \lambda f. b \circ f \circ a,$

powertype: $\mathbf{P} = \lambda a, b. \; \mathbf{V}(a \mid b)$

Types-as-closures

closures $\mathbf{I} \sqsubseteq a = a \circ a,$

$x : a \iff a\ x = x$

universe: $\mathbf{V} = \lambda a.\ \mathbf{I} \mid a \mid a \circ a \mid \ldots,$

exponentials: $a \to b = \lambda f. b \circ f \circ a,$

powertype: $\mathbf{P} = \lambda a, b.\ \mathbf{V}(a \mid b)$

products: Prod, unit, semi,

Types-as-closures

closures $\mathbf{I} \sqsubseteq a = a \circ a,$

$x : a \iff a\ x = x$

universe: $\mathbf{V} = \lambda a.\ \mathbf{I} \mid a \mid a \circ a \mid \ldots,$

exponentials: $a \to b = \lambda f. b \circ f \circ a,$

powertype: $\mathbf{P} = \lambda a, b.\ \mathbf{V}(a \mid b)$

products: Prod, unit, semi,

coproducts: Sum, nil, div,

Types-as-closures

closures $\quad \mathbf{I} \sqsubseteq a = a \circ a,$

$x : a \iff a\ x = x$

universe: $\quad \mathbf{V} = \lambda a.\ \mathbf{I} \mid a \mid a \circ a \mid \ldots,$

exponentials: $\quad a \to b = \lambda f. b \circ f \circ a,$

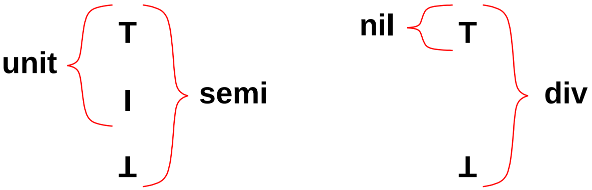
powertype: $\quad \mathbf{P} = \lambda a, b.\ \mathbf{V}(a \mid b)$

products: Prod, unit, semi,

coproducts: Sum, nil, div,

lower powerdomains: Sset

Dropped and Lifted types



nullary products

nullary coproducts

Simple types are definable

Theorem

For each simple type τ*,*
there is an SKJ-definable closure $[\tau]$ *s.t.*
for any SK-term x
x : τ *syntactically iff* $[\tau]\mathrm{x} = \mathrm{x}$ *mod* $\mathcal{H}^*$*.*

Simple types are definable

Theorem

For each simple type τ,
there is an SKJ-definable closure $[\tau]$ s.t.
for any SK-term x
x $: \tau$ *syntactically iff* $[\tau]$x $=$ x *mod* $\mathcal{H}^*$.

Conjecture

...for any SKJ-term x*...*

Trick: Close over section-retract pairs

$$\begin{aligned}
&\text{Simple } f = \mathbf{V}\ (\\
&\quad f\ \mathbf{I}\ \mathbf{I}\\
&\quad |\ f\ (\lambda x.\ x\ \top)\ (\lambda x, y.\ x)\\
&\quad |\ f\ (\lambda x.\ x\ \bot)\ (\lambda x, y.\ x \mid \text{div}\ y)\\
&\quad |\ \text{Simple}\lambda a, a'.\ \text{Simple}\lambda b, b'.\ f\ (a' \to b)\ (a \to b')\\
&)
\end{aligned}$$

e.g. booleans $\tau = a \to a \to a...$

$$\bot,\ \mathbf{K},\ \mathbf{F},\ \top\ :\ \text{Simple}\lambda a, a'.\ a \to a \to a'$$

Trick: Disambiguation

$$\mathbf{K} \mid \mathbf{F} \;:\; \mathsf{Simple}\,\lambda \mathsf{a}, \mathsf{a}'.\; \mathsf{a} \to \mathsf{a} \to \mathsf{a}'$$

Trick: Disambiguation

$$\mathbf{K} \mid \mathbf{F} \ : \ \text{Simple}\lambda a, a'.\ a \to a \to a'$$

$$\text{disambiguate} \ := \ \lambda q.\ q(q\ \bot\ \top)(q\ \top\ \bot)$$

Trick: Disambiguation

$$\mathbf{K} \mid \mathbf{F} \ : \ \mathrm{Simple}\lambda a, a'.\ a \to a \to a'$$

$$\begin{aligned} \text{disambiguate} \ &:= \ \lambda q.\ q(q\ \bot\ \top)(q\ \top\ \bot) \\ &:\ \bot, \mathbf{K}, \mathbf{F} \ \mapsto \ \bot \end{aligned}$$

Trick: Disambiguation

$$\mathbf{K} \mid \mathbf{F} \ : \ \text{Simple}\,\lambda a, a'.\ a \to a \to a'$$

$$\begin{aligned} \text{disambiguate} \ &:= \ \lambda q.\ q(q \perp \top)(q \top \perp) \\ &: \ \perp, \mathbf{K}, \mathbf{F} \ \mapsto \ \perp \\ &: \ \top, \mathbf{K} \mid \mathbf{F} \ \mapsto \ \top \end{aligned}$$

Feature: Problems of specific complexity

$x = \mathbf{I}, x = \mathbf{K}$ are Π^0_2-complete

Feature: Problems of specific complexity

$x = \mathbf{I}, x = \mathbf{K}$ are Π^0_2-complete

unit $x = \mathbf{I}$ is only Π^0_1

Feature: Problems of specific complexity

$x = \mathbf{I}, x = \mathbf{K}$ are Π^0_2-complete

unit $x = \mathbf{I}$ is only Π^0_1

semi $x = \mathbf{I}$ is only Δ^0_2

Feature: Problems of specific complexity

$x = \mathbf{I}$, $x = \mathbf{K}$ are Π^0_2-complete

unit $x = \mathbf{I}$ is only Π^0_1

semi $x = \mathbf{I}$ is only Δ^0_2

bool $x = \mathbf{K}$ is only Δ^0_1 in test_bool $x = \mathbf{I}$

$\mathcal{H}^*$ for randomness

$\mathcal{H}^*$ for randomness

Generalize to random convergence

$$\frac{\forall \underline{M}.\ \ x\ \ \underline{M}\ \ \text{conv} \quad \Longrightarrow \quad y\ \ \underline{M}\ \ \text{conv}}{\mathcal{H}^* \ \vdash \ x \sqsubseteq y}$$

$\mathcal{H}^*$ for randomness

Generalize to random convergence

$$\frac{\forall \underline{M}.\ \mathbb{P}[x\ \underline{M}\ \text{conv}] \ \leq\ \mathbb{P}[y\ \underline{M}\ \text{conv}]}{\mathcal{H}^* \ \vdash\ x \sqsubseteq y}$$

$\mathcal{H}^*$ for randomness

Generalize to random convergence

$$\frac{\forall \underline{M}.\ \mathbb{P}[x\ \ \underline{M}\ \ \text{conv}] \ \leq\ \mathbb{P}[y\ \ \underline{M}\ \ \text{conv}]}{\mathcal{H}^* \ \vdash\ x \sqsubseteq y}$$

Add random bit: $(x+y)z = x\ \ z + y\ \ z$

$\mathcal{H}^*$ for randomness

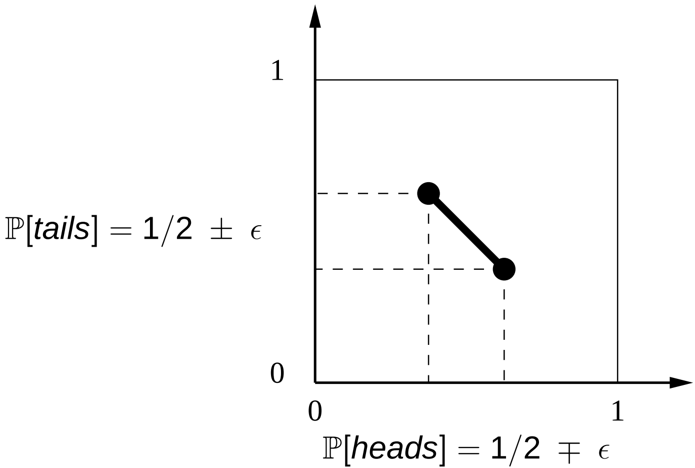
Generalize to random convergence

$$\frac{\forall \underline{M}.\ \mathbb{P}[x\ \underline{M}\ \text{conv}] \ \leq\ \mathbb{P}[y\ \underline{M}\ \text{conv}]}{\mathcal{H}^* \ \vdash\ x \sqsubseteq y}$$

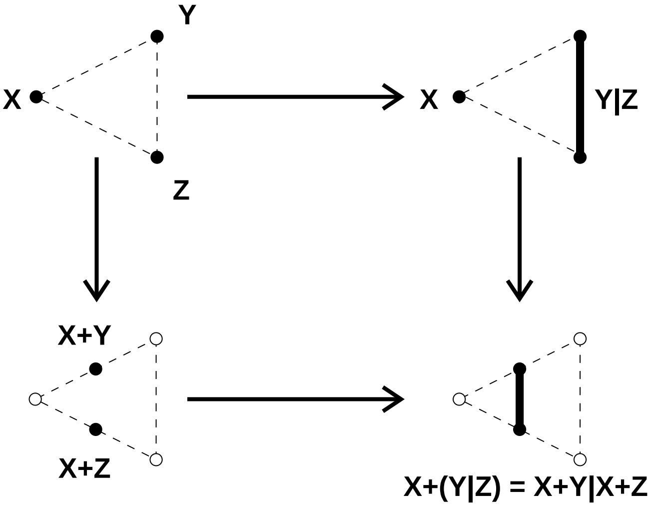
Add random bit: $(x + y)z = x\ z + y\ z$

$$\mathbb{P}[x + y\ \text{conv}] = \frac{\mathbb{P}[x\ \text{conv}] + \mathbb{P}[y\ \text{conv}]}{2}$$

Convex Sets of Probability Distributions



Mixture Distributes over Join



A Monad for CSPDs

lower powerdomain	probability
$\Box(\phi \to \psi) \to \Box\phi \to \Box\psi$	$\bigcirc(\phi \to \psi) \to \bigcirc\phi \to \bigcirc\psi$
$\phi \to \Box\phi$	$\phi \to \bigcirc\phi$
$\Box\Box\phi \to \Box\phi$	$\bigcirc\bigcirc\phi \to \bigcirc\phi$

A Monad for CSPDs

lower powerdomain	probability
$\Box(\phi \to \psi) \to \Box\phi \to \Box\psi$	$\bigcirc(\phi \to \psi) \to \bigcirc\phi \to \bigcirc\psi$
$\phi \to \Box\phi$	$\phi \to \bigcirc\phi$
$\Box\Box\phi \to \Box\phi$	$\bigcirc\bigcirc\phi \to \bigcirc\phi$

$$\bigcirc\Box\phi \to \Box\bigcirc\phi$$

A Monad for CSPDs

lower powerdomain	probability
$\Box(\phi \to \psi) \to \Box\phi \to \Box\psi$	$\bigcirc(\phi \to \psi) \to \bigcirc\phi \to \bigcirc\psi$
$\phi \to \Box\phi$	$\phi \to \bigcirc\phi$
$\Box\Box\phi \to \Box\phi$	$\bigcirc\bigcirc\phi \to \bigcirc\phi$

$$\bigcirc\Box\phi \to \Box\bigcirc\phi$$

$\Box\bigcirc$ = Fuzzy, for CSPDs

Lifting Church-style terms

$$\mathrm{lift}_{a,b} \ : \ (a \to \mathrm{Fuzzy}\ \ b) \ \to \ \mathrm{Fuzzy}\ \ a \to \mathrm{Fuzzy}\ \ b$$

Lifting Church-style terms

$$\text{lift}_{a,b} : (a \to \text{Fuzzy } b) \to \text{Fuzzy } a \to \text{Fuzzy } b$$

$$\text{lift}_{bool,b} \; f \; x = x(f \; \mathbf{K})(f \; \mathbf{F})$$

Lifting Church-style terms

$$\mathsf{lift}_{a,b} \ : \ (a \to \mathsf{Fuzzy}\ \ b) \ \to \ \mathsf{Fuzzy}\ \ a \to \mathsf{Fuzzy}\ \ b$$

$$\mathsf{lift}_{bool,b}\ \ f\ \ x = x(f\ \ \mathbf{K})(f\ \ \mathbf{F})$$

compatible with simple types

Loose end: SKRJ-definable simple types?

Disambiguation fails:

$$\begin{aligned}\text{disambiguate}(\mathbf{K}+\mathbf{F}) &= (\top+\bot) \;+\; (\bot+\top)\\ &= \bot+\top\\ &\neq \top\end{aligned}$$

Loose end: SKRJ-definable simple types?

Disambiguation fails:

$$\begin{aligned}\text{disambiguate}(\mathbf{K}+\mathbf{F}) &= (\top+\bot) \;+\; (\bot+\top)\\ &= \bot+\top\\ &\neq \top\end{aligned}$$

Section-retract pairs must be head-affine:

$$\begin{array}{rcll}(\lambda x.\ x\ \top) & , & (\lambda x,y.\ x) & \textit{ok}\\ (\lambda x,y.\ x\ y\ y) & , & (\lambda x,y,z.\ x(y \mid z)) & \textit{ok}\\ (\lambda x.\ x\ \bot) & , & (\lambda x,y.\ x \mid \text{div}\ y) & \textit{bad}\end{array}$$

e.g., $\mathbf{K} \mid \bot+\top = \mathbf{K}+\top$

Adding logical strength (SKJO)

Adding logical strength (SKJO)

ϕ : nat → bool is total

$$\forall x :: \text{test\_nat}.\ \phi\ x = \mathbf{K} \iff \bigsqcup_n\ \phi\ n\ \bot\ \mathbf{I} = \bot$$

Adding logical strength (SKJO)

$\phi : \text{nat} \to \text{bool}$ is total

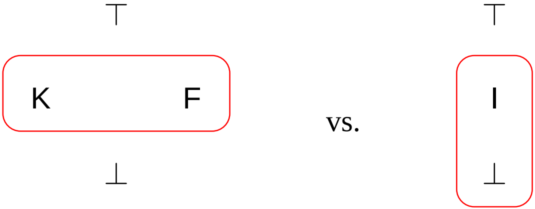
$$\forall x :: \text{test\_nat}.\ \phi\ x = \mathbf{K} \iff \bigsqcup_n\ \phi\ n\ \bot\ \mathbf{I} = \bot$$
$$\exists x :: \text{test\_nat}.\ \phi\ x = \mathbf{K} \iff \bigsqcup_n\ \phi\ n\ \mathbf{I}\ \bot = \mathbf{I}$$

Adding logical strength (SKJO)

ϕ : nat → bool is total

$$\forall x :: \text{test\_nat}.\ \phi\ x = \mathbf{K} \iff \bigsqcup_n \phi\ n\ \bot\ \mathbf{I} = \bot$$

$$\exists x :: \text{test\_nat}.\ \phi\ x = \mathbf{K} \iff \bigsqcup_n \phi\ n\ \mathbf{I}\ \bot = \mathbf{I}$$



A Π^1_1-complete semi-oracle

$$\mathbf{O}_{\mathbb{N}}\{\phi\}$$

A Π^1_1-complete semi-oracle

$$\mathbf{O}_{\mathbb{N}}\{\phi\} = \begin{cases} \top & \text{if } \exists \mathsf{n} :: \mathsf{test\_nat}.\ \phi\ \mathsf{n} = \top \\ & \\ & \end{cases}$$

A Π^1_1-complete semi-oracle

$$\mathbf{O}_{\mathbb{N}}\{\phi\} = \begin{cases} \top & \text{if } \exists \mathsf{n} :: \mathsf{test\_nat}.\ \phi\ \mathsf{n} = \top \\ \mathbf{I} & \text{if } \forall \mathsf{n} :: \mathsf{test\_nat}.\ \phi\ \mathsf{n} = \mathbf{I} \end{cases}$$

A Π^1_1-complete semi-oracle

$$\mathbf{O}_{\mathbb{N}}\{\phi\} = \begin{cases} \top & \text{if } \exists \mathsf{n} :: \mathsf{test\_nat}.\ \phi\ \mathsf{n} = \top \\ \mathbf{I} & \text{if } \forall \mathsf{n} :: \mathsf{test\_nat}.\ \phi\ \mathsf{n} = \mathbf{I} \\ \bot & \text{if } \exists \mathsf{n} :: \mathsf{test\_nat}.\ \phi\ \mathsf{n} = \bot \end{cases}$$

A Π^1_1-complete semi-oracle

$$\mathbf{O}_{\mathbb{N}}\{\phi\} = \begin{cases} \top & \text{if } \exists n :: \text{test\_nat}.\ \phi\ n = \top \\ \mathbf{I} & \text{if } \forall n :: \text{test\_nat}.\ \phi\ n = \mathbf{I} \\ \bot & \text{if } \exists n :: \text{test\_nat}.\ \phi\ n = \bot \end{cases}$$

$$\mathbf{O}_{\mathbb{N}} \quad := \quad \mathbf{O}\{\text{test\_nat}\}$$

A Π^1_1-complete semi-oracle

$$\mathbf{O}_{\mathbb{N}}\{\phi\} = \begin{cases} \top & \text{if } \exists n :: \text{test\_nat. } \phi \ n = \top \\ \mathbf{I} & \text{if } \forall n :: \text{test\_nat. } \phi \ n = \mathbf{I} \\ \bot & \text{if } \exists n :: \text{test\_nat. } \phi \ n = \bot \end{cases}$$

$$\mathbf{O}_{\mathbb{N}} \quad := \quad \mathbf{O}\{\text{test\_nat}\}$$

bool x $= \mathbf{K}$ is Δ^1_1 in test_bool x $= \mathbf{I}$

Comonadic codes: problem

Gödel codes take up space

$\mathbf{S}\ \mathbf{K} = \mathbf{K}\ \mathbf{I}$ but $\{\mathbf{S}\ \mathbf{K}\} \neq \{\mathbf{K}\ \mathbf{I}\}$

Comonadic codes: problem

Gödel codes take up space

$\mathbf{S}\ \mathbf{K} = \mathbf{K}\ \mathbf{I}$ but $\{\mathbf{S}\ \mathbf{K}\} \neq \{\mathbf{K}\ \mathbf{I}\}$

Ideal 1-1 coding operation

$$x = y \iff \{x\} = \{y\}$$

Comonadic codes: problem

Gödel codes take up space

$\mathbf{S}\ \mathbf{K} = \mathbf{K}\ \mathbf{I}$ but $\{\mathbf{S}\ \mathbf{K}\} \neq \{\mathbf{K}\ \mathbf{I}\}$

Ideal 1-1 coding operation

$$x = y \iff \{x\} = \{y\}$$

...but that is inconsistent...

Comonadic codes: solution

Quotient WRT provable equality

$$\Pr(x = y) \iff \{x\} = \{y\}$$

Comonadic codes: solution

Quotient WRT provable equality

$$\Pr(x = y) \iff \{x\} = \{y\}$$

Let $a, b : \mathbf{V}$, and $\Box a = \mathrm{Code}\{a\}$.

$$\begin{aligned} \mathrm{Apply}\{a\}\{b\} &: \Box(a \to b) \to \Box a \to \Box b \\ \mathrm{Eval}\{a\} &: \Box a \to a \\ \mathrm{Quote}\{a\} &: \Box a \to \Box\Box a \\ \mathrm{Code}\{a\}\{x\} &: \Box a \end{aligned}$$

Comonadic codes: solution

Quotient WRT provable equality

$$\Pr(x = y) \iff \{x\} = \{y\}$$

Let $a, b : \mathbf{V}$, and $\Box a = \text{Code}\{a\}$.

$$\begin{aligned}
\text{Apply}\{a\}\{b\} &: \Box(a \to b) \to \Box a \to \Box b \\
\text{Eval}\{a\} &: \Box a \to a \\
\text{Quote}\{a\} &: \Box a \to \Box\Box a \\
\text{Code}\{a\}\{x\} &: \Box a \\
\text{Fix}\{a\} &: \Box(\Box a \to a) \to \Box a
\end{aligned}$$

Implementation

Adapts Todd-Coxeter to SKJ, $\sqsubseteq$

Implementation

Adapts Todd-Coxeter to SKJ, $\sqsubseteq$

Reasons with support A,

$$\sqsubseteq \subseteq A \times A$$
$$(- \ \ -) : A \times A \to A$$

Implementation

Adapts Todd-Coxeter to SKJ, $\sqsubseteq$

Reasons with support A,

$$\sqsubseteq \subseteq A \times A$$
$$(-\ \ -) : A \times A \to A$$

Saturation complexity: $(|A|^2 \text{ space}) \times (|A|^3 \text{ time})$

Implementation

Adapts Todd-Coxeter to SKJ, $\sqsubseteq$

Reasons with support A,

$$\sqsubseteq \subseteq A \times A$$
$$(-\ \ -) : A \times A \to A$$

Saturation complexity: $(|A|^2$ space)$\times$($|A|^3$ time)

1 day, $|A|$=10k, 5% application + 95% order known

Translating SK$\leftrightarrow\lambda$-**let**-calculus: problem

Translating SK$\leftrightarrow\lambda$-**let**-calculus: problem

Easy to reason about

$$\frac{}{\mathbf{S}\ x\ y\ z = x\ z(y\ z)} \qquad \frac{}{\mathbf{K}\ x\ y = x} \qquad \frac{f = g \qquad x = y}{f\ x = g\ y}$$

Translating SK↔λ-**let**-calculus: problem

Easy to reason about

$$\frac{}{\mathbf{S}\ x\ y\ z = x\ z(y\ z)} \qquad \frac{}{\mathbf{K}\ x\ y = x} \qquad \frac{f = g \qquad x = y}{f\ x = g\ y}$$

Easy to write

$\mathbf{let}\ s := (\lambda n, f, x.\ f(n\ f\ x)).\quad \mathbf{let}\ z := (\lambda -, x.\ x).\quad s(s(s(z)))$

$\mapsto$

$\mathbf{S}(\mathbf{S}(\mathbf{K}\ \mathbf{S})\mathbf{K})(\mathbf{S}(\mathbf{S}(\mathbf{K}\ \mathbf{S})\mathbf{K})(\mathbf{S}\ \mathbf{K}\ \mathbf{K}))(\mathbf{S}(\mathbf{S}(\mathbf{K}\ \mathbf{S})\mathbf{K}))(\mathbf{S}\ \mathbf{K})$

Translating SK↔λ-**let**-calculus: problem

Easy to reason about

$$\frac{}{\mathbf{S}\ x\ y\ z = x\ z(y\ z)} \qquad \frac{}{\mathbf{K}\ x\ y = x} \qquad \frac{f = g \qquad x = y}{f\ x = g\ y}$$

Easy to write

$\textbf{let}\ s := (\lambda n, f, x.\ f(n\ f\ x)).\quad \textbf{let}\ z := (\lambda -, x.\ x).\quad s(s(s(z)))$

$\mapsto$

$\mathbf{S}(\mathbf{S}(\mathbf{K}\ \mathbf{S})\mathbf{K})(\mathbf{S}(\mathbf{S}(\mathbf{K}\ \mathbf{S})\mathbf{K})(\mathbf{S}\ \mathbf{K}\ \mathbf{K}))(\mathbf{S}(\mathbf{S}(\mathbf{K}\ \mathbf{S})\mathbf{K}))(\mathbf{S}\ \mathbf{K})$

Impossible to read

Translating SK↔λ-**let**-calculus: solution

(1) find simple/simplest I,K,F,B,C,W,S-term

$$
\begin{aligned}
&\mathbf{S}(\mathbf{S}(\mathbf{K}\ \mathbf{S})\mathbf{K})(\mathbf{S}(\mathbf{S}(\mathbf{K}\ \mathbf{S})\mathbf{K})(\mathbf{S}\ \mathbf{K}\ \mathbf{K}))(\mathbf{S}(\mathbf{S}(\mathbf{K}\ \mathbf{S})\mathbf{K}))(\mathbf{S}\ \mathbf{K})\\
&\quad = \ \mathbf{S}\ \mathbf{B}(\mathbf{W}\ \mathbf{B})(\mathbf{S}\ \mathbf{B})\mathbf{F}.
\end{aligned}
$$

Translating SK$\leftrightarrow\lambda$-**let**-calculus: solution

(1) find simple/simplest I,K,F,B,C,W,S-term

$$\mathbf{S}(\mathbf{S}(\mathbf{K}\ \mathbf{S})\mathbf{K})(\mathbf{S}(\mathbf{S}(\mathbf{K}\ \mathbf{S})\mathbf{K})(\mathbf{S}\ \mathbf{K}\ \mathbf{K}))(\mathbf{S}(\mathbf{S}(\mathbf{K}\ \mathbf{S})\mathbf{K}))(\mathbf{S}\ \mathbf{K}) = \mathbf{S}\ \mathbf{B}(\mathbf{W}\ \mathbf{B})(\mathbf{S}\ \mathbf{B})\mathbf{F}.$$

(2) decompile into λ-let-calculus

$$\mathbf{S}\ \mathbf{B}(\mathbf{W}\ \mathbf{B})(\mathbf{S}\ \mathbf{B})\mathbf{F} \mapsto \mathbf{let}\ \ a := (\lambda b, c, d.\ \ c(b\ \ c\ \ d)).\quad a(a(a\lambda e, f.\ \ f))$$

Translating SK$\leftrightarrow\lambda$-**let**-calculus: solution

(1) find simple/simplest I,K,F,B,C,W,S-term

$$\mathbf{S(S(K\ S)K)(S(S(K\ S)K)(S\ K\ K))(S(S(K\ S)K))(S\ K)} \\ = \mathbf{S\ B(W\ B)(S\ B)F}.$$

(2) decompile into λ-let-calculus

$$\mathbf{S\ B(W\ B)(S\ B)F} \\ \mapsto \mathbf{let}\ a := (\lambda b, c, d.\ c(b\ c\ d)).\quad a(a(a\lambda e, f.\ f))$$

- affine-β-η-reduce whenever possible
- combine copied subexpressions via **let**-in
- be generous with variable names;
  e.g. favor $\lambda a.\lambda b.b$ over $\lambda a.\lambda a.a$

Automated conjecturing

Problem: $\mathcal{H}^*$ is Π^0_2-complete.

Automated conjecturing

Problem: $\mathcal{H}^*$ is Π^0_2-complete.

Solution: Mine for missing equations.

Automated conjecturing

Problem: $\mathcal{H}^*$ is Π^0_2-complete.

Solution: Mine for missing equations.

(1) Define probability distribution over inequality proofs.

Automated conjecturing

Problem: $\mathcal{H}^*$ is Π^0_2-complete.

Solution: Mine for missing equations.

(1) Define probability distribution over inequality proofs.

(2) Propagate evidence of unprovability among $\sim 10^8$ pairs.

Automated conjecturing

Problem: $\mathcal{H}^*$ is Π_2^0-complete.

Solution: Mine for missing equations.

(1) Define probability distribution over inequality proofs.

(2) Propagate evidence of unprovability among $\sim 10^8$ pairs.

(3) Find $\sim$ 100 pairs with most evidence.

Learning an optimal basis

Problem: sensitivity to basis weights

Learning an optimal basis

Problem: sensitivity to basis weights

$$\{\text{app}@0.5,\ \ \mathbf{S}@0.3,\ \ \mathbf{K}@0.2\}$$

–versus–

$$\{\text{app}@0.6,\ \ \mathbf{S}@0.2, \mathbf{K}@0.2\}$$

Learning an optimal basis

Problem: sensitivity to basis weights

$$\{\text{app@0.5}, \ \ \mathbf{S}\text{@0.3}, \ \ \mathbf{K}\text{@0.2}\}$$

–versus–

$$\{\text{app@0.6}, \ \ \mathbf{S}\text{@0.2}, \mathbf{K}\text{@0.2}\}$$

Solution: Locally minimize corpus complexity

Learning an optimal basis

Problem: sensitivity to basis weights

$$\{\text{app}@0.5,\ \ \mathbf{S}@0.3,\ \ \mathbf{K}@0.2\}$$

–versus–

$$\{\text{app}@0.6,\ \ \mathbf{S}@0.2, \mathbf{K}@0.2\}$$

Solution: Locally minimize corpus complexity

Empirically most useful terms:

B, **C**, **K**, (**C I**), **I**, **S**, **J**, **V**, **Y**, (**C B**), **F**, **P**, **W**

Loose End: Better Verification

Loose End: Better Verification

integrate with Knuth-Bendix

Loose End: Better Verification

integrate with Knuth-Bendix

additional knowledge representations

$$\circ : A \times A \longrightarrow A$$

$$| : A \times A \longrightarrow A$$

Loose End: Better Verification

integrate with Knuth-Bendix

additional knowledge representations

$$\circ : A \times A \to A$$

$$| : A \times A \to A$$

embarrassingly parallel rule search

Thanks to:

Rick Statman, Dana Scott,

Kevin Kelly, and James Cummings

Thanks to:

Rick Statman, Dana Scott,
Kevin Kelly, and James Cummings

Questions?