Definable (types-as-)closures
in concurrent combinatory algebra

Fritz Obermeyer

Department of Mathematics
Carnegie-Mellon University

2007:03:27

Outline

Motivation: complexity-of-definition

What mathematical objects are definable?
$\rightarrow$ First, the constructive objects are definable,
say r.e. sets mod r.e. relations.

How simple is a given object?
$\rightarrow$ As simple as its shortest description. (Kolmogorov)

When is a description simple?
$\rightarrow$ Fewer and shorter symbols (parametrized)

How does complexity depend on language (parameters)?
Hmm... these are more than r.e. sets...
these are weighted grammars or weighted presentations.
Simpler grammars/signatures are simpler to parametrize.

Why untyped λ-calculus?

Church: One language for r.e. sets mod r.e. relations is...
λ-calculus mod an r.e. theory: β, $\beta{+}\eta$, etc.

Curry: λ-calculus has a complicated syntax, but
a few closed terms generate all others.

What basis is sufficient? Terms need to:

- do nothing at all: $\lambda \mathsf{f}.\ \mathsf{f}$
- move terms around: $\lambda \mathsf{f}, \mathsf{x}, \mathsf{y}.\ \mathsf{f}(\mathsf{x}\ \mathsf{y})$, $\lambda \mathsf{f}, \mathsf{x}, \mathsf{y}.\ (\mathsf{f}\ \mathsf{x})\mathsf{y}$
- project/ignore terms: $\lambda \mathsf{f}, \mathsf{x}.\ \mathsf{f}$
- copy terms: $\lambda \mathsf{f}, \mathsf{x}.\ \mathsf{f}\ \mathsf{x}\ \mathsf{x}$
- (join terms: $\lambda \mathsf{f}, \mathsf{g}.\ \mathsf{f} \mid \mathsf{g}$?)

That's enough! Many other bases would work.

Why semantics, extensionality?

How simple is a given object?
$\rightarrow$ should not depend on any one description, but also
$\rightarrow$ should not depend on any one language

We want meaning, not syntax.
$\rightarrow$ Coarser theories/quotients are better.
$\rightarrow$ But "empty" / "undefined" should remain the same.

(1) Identify all "empty" / "undefined" terms (the theory $\mathcal{H}$)

(2) Then identify as much as consistently possible ($\mathcal{H}^*$)

From seqential λ-calculus, we get term fragment of $\mathcal{D}_\infty$.
From concurrent λ-calculus, we get term+join fragment.

But $\mathcal{H}^*$ is Π^0_2-complete, not r.e.; what about "constructive"?
$\rightarrow$ Approximate $\mathcal{H}^*$ by r.e. theory, e.g., ZFC.

Why concurrency (join)?

How are open/r.e. sets represented in λ-calc.?
$\rightarrow$ enumerations : nat $\rightarrow$ a, intersection, union
$\rightarrow$ semipredicates : a $\rightarrow \{\perp, \mathbf{I}\}$, intersection, no union

Disjunction is representable at meta-level (simulation)
but this does not work for oracles.
Add join as a primitive: a $\rightarrow$ semi becomes a lattice,

Scott came from opposite direction:
Some top. spaces yield models of λ-calculus
and in these models join is of course definable.
But $\mathcal{D}_\infty$,$\mathcal{P}\omega$ introduce extra junk, e.g. step functions.

So... Consider pure fragement of $\mathcal{D}_\infty$:
$=$ concurrent combinatory algebra, mod $\mathcal{H}^*$

Combinatory algebra with join

Combinatory algebra: equational, $\mathbf{S}$ and $\mathbf{K}$ for abstraction

$$\mathbf{K}\ \mathsf{x}\ \mathsf{y} = \mathsf{x} \qquad\qquad \mathbf{S}\ \mathsf{x}\ \mathsf{y}\ \mathsf{z} = \mathsf{x}\ \mathsf{z}(\mathsf{y}\ \mathsf{z})$$

Concurrent CA: partially ordered, also $\mathbf{J}$ for join

$$\mathbf{J}\ \mathsf{x}\ \mathsf{y} \sqsupseteq \mathsf{x} \qquad \mathbf{J}\ \mathsf{x}\ \mathsf{y} \sqsupseteq \mathsf{y} \qquad \frac{\mathsf{x} \sqsubseteq \mathsf{z} \qquad \mathsf{y} \sqsubseteq \mathsf{z}}{\mathbf{J}\ \mathsf{x}\ \mathsf{y} \sqsubseteq \mathsf{z}}$$

In either case, add $\top$ for error: $\quad \top\ \mathsf{x} = \mathsf{x}$, or $\top \sqsupseteq \mathsf{x}$

Translation from λ-calculus with join

$$\begin{aligned} [\![\lambda \mathsf{x}.\mathsf{M}]\!] &= \mathbf{K}\ \mathsf{M} && \mathsf{x}\ \textit{not free in}\ \mathsf{M} \\ [\![\lambda \mathsf{x}.\mathsf{M}\ \mathsf{N}]\!] &= \mathbf{S}[\![\lambda \mathsf{x}.\mathsf{M}]\!][\![\lambda \mathsf{x}.\mathsf{N}]\!] \\ [\![\mathsf{M} \mid \mathsf{N}]\!] &= \mathbf{J}[\![\mathsf{M}]\!][\![\mathsf{N}]\!] \end{aligned}$$

The completed term model

Definition

A term x converges iff $\exists n \in \mathbb{N}.\ x\ \top^{\sim n} = \top$.

Definition

(Scott's information ordering)
$\mathcal{H}^* \vdash x \sqsubseteq y$ iff $\forall M \in \langle \mathbf{S}, \mathbf{K} \rangle.$ M x conv $\implies$ M y conv.

Consider the term model $\mathcal{B}$ mod $\mathcal{H}^*$, with arbirary joins

$$\frac{}{\mathbf{S}, \mathbf{K}, \mathbf{J} \in \mathcal{B}} \qquad \frac{x \in \mathcal{B} \qquad y \in \mathcal{B}}{(x\ y) \in \mathcal{B}} \qquad \frac{X \subseteq \mathcal{B}}{\bigsqcup X \in \mathcal{B}}$$

Theorem

$\mathcal{B}$ is an algebraic lattice with join $\mathbf{J} = \mathbf{K} \mid \mathbf{S}\ \mathbf{K}$, bottom $\bot = \mathbf{Y}\ \mathbf{K}$, and top $\top = \mathbf{Y}\ \mathbf{J}$.

Semantically typed λ-calculus (sequential)

Types as idempotents: unityped $\longrightarrow$ typed

$$\begin{aligned} \mathsf{a}\ \text{type} &\iff \mathsf{a}\circ\mathsf{a} = \mathsf{a} \\ \mathsf{x}:\mathsf{a} &\iff \mathsf{a}\ \mathsf{x} = \mathsf{x} \\ \mathsf{f}:\mathsf{a}\to\mathsf{b} &\iff \mathsf{b}\circ\mathsf{f} = \mathsf{f} = \mathsf{f}\circ\mathsf{a} \end{aligned}$$

Is this really typed, fully abstract? Not usefully.

Range property: every range is a singleton or infinite.
$\Longrightarrow$ No booleans, $\Longrightarrow$ no numerals.

How many denotations of $\mathsf{not} : \mathbf{K}, \mathbf{F} \mapsto \mathbf{F}, \mathbf{K}$?
Infinitely many.

Is there a solution to $\mathsf{and}\ \mathsf{x}\ \mathsf{y} = \mathsf{and}\ \mathsf{y}\ \mathsf{x}$? No.

Semantically typed λ-calculus (concurrent)

Types as closures

$$\mathsf{a}\ \text{type} \iff \mathsf{a}\circ\mathsf{a} = \mathsf{a} \ \sqsupseteq \ \mathbf{I}$$

Is this really typed, fully abstract? Yes!

Range property fails, e.g. rng$(\mathbf{Y}(\mathbf{I} \mid \langle\top\rangle)) = \{\bot, \top\}$.
We will define bool, nat, ... from only $\mathbf{S}, \mathbf{K}, \mathbf{J}$.

How many denotations of 'not' ?
Still infinitely many solutions,
but unique maximum denotation.

Is there a solution to and x y = and y x ? Yes.

Main difference: coproducts (dropped, lifted)

What types are definable?

$$
\begin{array}{rcll}
x : a & \iff & a\ x = x & \textit{(inhabitation)} \\
\lambda x : a.M & = & (\lambda x.M) \circ a & \textit{(typed abstraction)} \\
 & & & \\
a \to b & = & \lambda f.b \circ f \circ a & \textit{(function = exponential)} \\
\forall y : a.M & = & \lambda x, y : a.M(x\ y) & \textit{(dependent, polymorphic)} \\
 & & & \\
a <: b & \iff & b \circ a \circ b = a & \textit{(subtyping)} \\
a \wedge b & = & \mathsf{Sub}\ a\ b & \textit{(type intersection)} \\
 & & & \\
a \times b & = & \mathsf{Prod}\ a\ b & \textit{(dropped products)} \\
a + b & = & \mathsf{Sum}\ a\ b & \textit{(dropped lifted sums)} \\
\exists y : a.M & = & \mathsf{Exists}\ \lambda y : a.M & \textit{(d'd l'd indexed sums)}
\end{array}
$$

Also atoms: type, any, nil, unit, bool, nat

The universal type of types

$$\mathsf{type} \;:=\; \lambda \mathsf{a}.\ \mathbf{I} \mid \mathsf{a} \mid \mathsf{a}\circ\mathsf{a} \mid \mathsf{a}\circ\mathsf{a}\circ\mathsf{a} \mid \dots \qquad = \; \lambda \mathsf{a}.\mathbf{Y}\lambda \mathsf{b}.\ \mathbf{I} \mid \mathsf{a}\circ\mathsf{b}$$

Theorem

type *is a closure, and* a : type $\iff$ a *is a closure.*

Proof.

(closure) type $\sqsupseteq \lambda \mathsf{a}.\mathsf{a} = \mathbf{I}$, and type(type a) = type a.

($\Longrightarrow$) Suppose a : type, i.e., $\mathsf{a} = \mathbf{I} \mid \mathsf{a} \mid \mathsf{a}\circ\mathsf{a} \mid \dots$.
Then $\mathsf{a} \sqsupseteq \mathbf{I}$, and $\mathsf{a}\circ\mathsf{a} = \mathsf{a}$.

($\Longleftarrow$) If $\mathsf{a} \sqsupseteq \mathbf{I}$ then $(\mathbf{I} \mid \mathsf{a} \mid \mathsf{a}\circ\mathsf{a} \mid \dots) = (\mathsf{a} \mid \mathsf{a}\circ\mathsf{a} \mid \dots)$.
If also $\mathsf{a} = \mathsf{a}\circ\mathsf{a}$, the chain collapses to a.

□

Maximal and minimal types

Everything is fixed by the identity, so the largest type is

$$\text{any} := \mathbf{I} \quad = \text{type } \mathbf{I}.$$
$$\text{inhab(any)} = \mathcal{B}$$

Every type is inhabited by $\top$, so the smallest type is

$$\text{nil} := \top \quad = \text{type } \top.$$
$$\text{inhab(nil)} = \{\top\}$$

nil is: terminal object, dropped initial object.
($\mathcal{B}$ has no initial object)

Function types (exponentials)

Definition

For any terms a, b, define the conjugation operator

$$\mathsf{a} \to \mathsf{b} \;:=\; \lambda \mathsf{f}.\mathsf{b} \circ \mathsf{f} \circ \mathsf{a} \qquad = \lambda \mathsf{f}, \mathsf{x}.\ \mathsf{b}(\mathsf{f}(\mathsf{a}\ \mathsf{x}))$$

(associates to the right)

Now define a binary operation on types

$$\begin{aligned} \mathsf{Exp} \;&:=\; \mathsf{type} \to \mathsf{type} \to \mathsf{type}\ (\lambda \mathsf{a}, \mathsf{b}.\ \mathsf{a} \to \mathsf{b}). \\ &=\; \lambda \mathsf{a} : \mathsf{type}.\ \mathsf{type} \to \mathsf{type}\ \lambda \mathsf{b}.\ \mathsf{a} \to \mathsf{b} \\ &=\; \lambda \mathsf{a} : \mathsf{type},\ \mathsf{b} : \mathsf{type}.\ \mathsf{type}\ (\mathsf{a} \to \mathsf{b}) \end{aligned}$$

We'll use this form often:

$$\mathsf{some\_term} \;:=\; \mathsf{its\_type}\ \ \mathsf{untyped\_definition}.$$

Navigating a minefield.

Consider the section-retract pair

$$\begin{aligned} \mathsf{R}_{mn} &:= \lambda f, w_1, \ldots, w_m, x, y_1, \ldots, y_n.\ f\ x \\ \mathsf{L}_{mn} &:= \lambda g, x.\ g\ \ \top^{\sim m}\ \ x\ \ \top^{\sim n} \end{aligned}$$

so that

$$\begin{aligned} &\mathsf{L}_{mn} \circ \mathsf{R}_{mn} = \mathbf{I} \\ &\mathsf{R}_{mn} \circ \mathsf{L}_{mn} = \lambda f, \underline{w}, x, \underline{y}.\ f\ \top^{\sim m}\ x\ \top^{\sim n}\ \sqsupseteq\ \mathbf{I} \end{aligned}$$

Hence $\mathsf{R}_{mn} \circ \mathsf{L}_{mn}$ is a closure.

(often omit indices: $\mathsf{L} \circ \mathsf{R} = \mathbf{I}$)

Think of L as a minefield of errors
and R as a map through the minefield.

Avoiding errors.

The Church numeral $1 = \lambda f, x.f\ x$ has simple type

$$(a \to a) \to a \to a \qquad \textit{note variance of each } a$$

Consider the action of $(L \to R) \to R \to L$ on 1:

$$\begin{aligned}
&(L \to R) \to R \to L\ (\lambda f, x.f\ x)\ f\ x \\
&\quad = \ L\ (\ (\lambda f, x.f\ x)\ (R \circ f \circ L)\ (R\ x)\ ) \\
&\quad = \ L\ (R \circ f \circ L\ (R\ x)) \\
&\quad = \ (L \circ R) \circ f \circ (L \circ R)\ x \\
&\quad = \ \mathbf{I} \circ f \circ \mathbf{I}\ x \\
&\quad = \ f\ x \quad = 1\ f\ x
\end{aligned}$$

Hence $1 : (L \to R) \to R \to L$.

actually works for any section-retract pair

Failing to avoid errors.

What about non-Church numerals, e.g., $\lambda f, x.x\ f$?

$$\begin{aligned}(L\to R)\to R\to L\ (\lambda f, x.x\ f)\ f\ x & \\ &= L\ (\ (\lambda f, x.x\ f)\ (R\circ f\circ L)\ (R\ x)\ ) \\ &= L\ (R\ x\ (R\circ f\circ L)) \\ &= L\ x \\ &= x\ \top \qquad \neq\ x\ f\end{aligned}$$

oops: $\lambda f, x.x\ f\ \div\ (L\to R)\to R\to L.$

Can we force any incorrect term up to $\top =$ error?
Can we raise any partial term up to a fixedpoint?
...sometimes...

The type of divergent computations

$$\mathsf{div} \;:= \bigsqcup \mathsf{m} \geq 0.\ \mathsf{L}_{\mathsf{m0}} \;= \bigsqcup \mathsf{m} \geq 0.\ \mathsf{m}\ \langle\top\rangle \quad = \mathsf{type}\ \langle\top\rangle$$

Theorem

inhab(div) $= \{\bot, \top\}$.

Proof.

Since $\bot\ \top = \bot$, $\quad \bot$: div.

Any other term q : div in question must converge

(recall q converges iff for some m, q $\top^{\sim \mathsf{m}} \equiv \top$).

Then

$$\begin{aligned} \mathsf{q} &= \mathsf{div}\ \mathsf{q} \\ &= \bigsqcup \mathsf{m} \geq 0.\ \mathsf{m}\ \langle\top\rangle\ \mathsf{q} \\ &= \bigsqcup \mathsf{m} \geq 0.\ \mathsf{q}\ \top^{\sim \mathsf{m}} \qquad = \top \end{aligned}$$

□

Moral: every candidate q stepped on a mine somewhere.

Protecting terms from divergence

We'll also need to make terms temporarily inert

$$\begin{aligned} \mathsf{curry} &:= \lambda \mathsf{f}, \mathsf{x}, \mathsf{y}.\ \mathsf{f}\langle \mathsf{x}, \mathsf{y}\rangle && = \lambda \mathsf{f}, \mathsf{x}, \mathsf{y}.\ \mathsf{f}(\lambda \mathsf{g}.\mathsf{g}\ \mathsf{x}\ \mathsf{y}) \\ \mathsf{uncurry} &:= \lambda \mathsf{g}, \langle \mathsf{x}, \mathsf{y}\rangle.\ \mathsf{g}\ \mathsf{x}\ \mathsf{y} && = \lambda \mathsf{g}, \mathsf{p}.\mathsf{p}\ \mathsf{g} \end{aligned}$$

Then

$$\begin{aligned} &\mathsf{uncurry} \circ \mathsf{curry} = \mathbf{I} \\ &\mathsf{curry} \circ \mathsf{uncurry} \ \sqsubseteq\!\!\!\!\!\!\!\!\!\;\;\ \mathbf{I} && \textit{(enough)} \end{aligned}$$

For example is $\mathsf{q} = \lambda \mathsf{f}, \_.\mathsf{f}(\mathsf{f}\ \bot)\ :\ \mathsf{a} \to \mathsf{a}$?
How do we see the second f without diverging?

$$\begin{aligned} \mathsf{C} \to \mathsf{U}\ \mathsf{q} &= \lambda \mathsf{f}.\ \mathsf{U}\ (\lambda \_.\ \mathsf{C}\ \mathsf{f}\ (\mathsf{C}\ \mathsf{f}\ \bot)) \\ &= \lambda \mathsf{f}.\ (\mathsf{U} \circ \mathsf{C})\ \lambda \_.\mathsf{f}\ (\mathsf{C}\ \mathsf{f}\ \bot) \\ &= \lambda \mathsf{f}, \_.\ \mathsf{f}\ (\lambda \mathsf{x}.\ \mathsf{f}\ \langle \bot, \mathsf{x}\rangle) \end{aligned}$$

Closing this operation: type $\mathsf{C} \to \mathsf{U}\ \mathsf{q} = \lambda \mathsf{f}, \_.\mathsf{f}\ \top$.

Constructing simple concurrent types

Generalize to functors of mixed variance:
join over all sorts of section-retract pairs.

$$\begin{aligned}\mathsf{Simple} \;:=\; & \mathsf{any}\!\rightarrow\!\mathsf{type}\ (\\ & \quad \lambda \mathsf{f}.\ \mathsf{f}\ \mathsf{curry}\ \mathsf{uncurry}\\ & \quad\quad \mid \textstyle\bigsqcup \mathsf{m},\mathsf{n} \geq 0.\ \mathsf{f}\ \mathsf{R_{mn}}\ \mathsf{L_{mn}}\\ &).\end{aligned}$$

alternate definition

For example

$$\begin{aligned}&\mathsf{div} = \mathsf{Simple}\ \lambda \mathsf{a}, \mathsf{a}'.\ \mathsf{a}'\\ &\mathsf{nat}\ \ <:\ \ \mathsf{Simple}\ \lambda \mathsf{a}, \mathsf{a}'.\ (\mathsf{a}' \!\rightarrow\! \mathsf{a}) \!\rightarrow\! \mathsf{a} \!\rightarrow\! \mathsf{a}'\\ &\mathsf{Prod}\ \ <:\ \ \lambda \mathsf{a}\!:\!\mathsf{type}, \mathsf{b}\!:\!\mathsf{type}.\ \mathsf{Simple}\ \lambda \mathsf{c}, \mathsf{c}'.\ (\mathsf{a} \!\rightarrow\! \mathsf{b} \!\rightarrow\! \mathsf{c}) \!\rightarrow\! \mathsf{c}'\end{aligned}$$

This is amost enough, but there may be $\top$'s in the body.

Checking the body for errors

We saw $(\mathsf{Simple}\ \lambda \mathsf{a}, \mathsf{a}'.\ \mathsf{a} \to \mathsf{a}')\ \lambda \mathsf{f}, \_.\mathsf{f}(\mathsf{f}\ \bot)\ =\ \lambda \mathsf{f}, \_.\mathsf{f}\ \top$.
But is $\lambda \mathsf{f}, \_.\mathsf{f}\ \top\ :\ \mathsf{a} \to \mathsf{a}$?

Try combining intro and elim forms: $\lambda \mathsf{x}.\mathsf{x}\ \mathbf{I} = \langle \mathbf{I} \rangle$.

$$\langle \mathbf{I} \rangle\ (\lambda \mathsf{f}, \_.\ \mathsf{f}\ \top)\ =\ \lambda \mathsf{x}.\ \mathbf{I}\ \top\qquad =\ \top$$

What about numerals? Is $\lambda \mathsf{f}, \_.\mathsf{f}(\ldots(\mathsf{f}\ \top)\ldots)\ :\ \mathsf{nat}$?
Try intro and elim forms: $\lambda \mathsf{n}.\mathsf{n}\ \mathsf{succ}\ \mathsf{zero} = \langle \mathsf{succ}, \mathsf{zero} \rangle$.

$$\langle \mathsf{s}, \mathsf{z} \rangle\ \lambda \mathsf{f}, \_.\mathsf{f}(\ldots(\mathsf{f}\ \top)\ldots) = \mathsf{s}(\ldots(\mathsf{s}\ \top)\ldots)\ =\ \top$$

This is enough: descend into body with intro and elim forms.

Intermezzo: concurrent head normal form

Definition

A head normal form is a λ-term

$$\lambda \mathsf{x}_1, \ldots, \mathsf{x}_\mathsf{v}.\ \mathsf{x}\ \mathsf{M}_1\ \ldots\ \mathsf{M}_\mathsf{a}$$

where $\mathsf{a}, \mathsf{v} \geq 0$, and $\mathsf{M}_1, \ldots, \mathsf{M}_\mathsf{a}$ are concurrent λ-terms.
Call x the head variable, and $\mathsf{M}_1, \ldots, \mathsf{M}_\mathsf{a}$ the body.

Definition

A concurrent Böhm tree is a h.n.f.
where the M′s are recursively joins of BT's.

Proposition

Everything is a join of h.n.f.s (modulo observability $\mathcal{H}^*$).

E.g. $\mathbf{J} = \lambda \mathsf{x}, \mathsf{y}.\mathsf{x} \mid \mathsf{y}\ = (\lambda \mathsf{x}, \mathsf{y}.\mathsf{x}) \mid (\lambda \mathsf{x}, \mathsf{y}.\mathsf{y})$, by η-conversion.

Intermezzo: interpolation by head normal forms

Proposition

Everything is a join of h.n.f.s (modulo observability $\mathcal{H}^*$).

- Necessary for $\mathbf{S}, \mathbf{K}, \mathbf{J}$-definable closures.
- Fails in Scott's model: step functions.
- Trivially true in the completed term model $\mathcal{B}$.

Corollary

If q *converges then* q *extends a h.n.f.;*

If $\mathsf{q} \not\sqsubseteq \mathsf{q}'$ *then* $\mathsf{q} \sqsupseteq \mathsf{M} \not\sqsubseteq \mathsf{q}'$ *for a h.n.f.* M.

...and now for the Main Example...

Can't say no? maybe you need a...

$$\mathsf{semi} \;:=\; \mathsf{type}\ ((\mathsf{Simple}\ \lambda a, a'.\ a \to a') \;\mid\; \langle \mathbf{I} \rangle).$$

Theorem

$\mathsf{inhab}(\mathsf{semi}) = \{\bot, \mathbf{I}, \top\}$.

Proof.

$\bot : \mathsf{semi}$ by β-reduction. Any other $q : \mathsf{semi}$ converges, say

$$q \;\sqsupseteq\; q' = \lambda f, x_1, \ldots, x_n.\ z\ M_1\ \ldots\ M_m$$

Show that either $q = \top$ or

$z = f, \quad m = n$ *(use minefields)*,

$M_i \sqsubseteq x_i$ *(descend, minefields, curry)*.

Finally raise q' up to $\mathbf{I}$ with minefields. ▸ details

□

Enforcing sequentiality

Consider a bad definition of bool
$\mathsf{boool} \;:=\; \mathsf{type}\ ((\mathsf{Simple}\ \lambda \mathsf{a}, \mathsf{a}'.\ \mathsf{a} \to \mathsf{a} \to \mathsf{a}') \;\mid\; \langle \mathbf{K}, \mathbf{F} \rangle).$

Theorem

$\mathsf{inhab}(\mathsf{boool}) = \{\bot, \mathbf{K}, \mathbf{F}, \mathbf{J}, \top\}$. *(recall* $\mathbf{J} = \mathbf{K} \mid \mathbf{F}$*)*

Proof.

Similar to semi, but now q can extend two h.n.f.'s:

$$\mathsf{q} \;\sqsupseteq\; \lambda \mathsf{x}, \mathsf{y}.\mathsf{x} = \mathbf{K}, \qquad \mathsf{q} \;\sqsupseteq\; \lambda \mathsf{x}, \mathsf{y}.\mathsf{y} = \mathbf{F}$$

$\mathbf{J}$ extends both.

$\square$

How to ensure sequentiality? Make q decide.

Corrected definition of bool

$$\text{make\_up\_your\_mind} \;:=\; \lambda \text{q. q} \perp (\text{q} \top \perp).$$
$$\text{bool} \;:=\; \text{type (boool} \mid \text{make\_up\_your\_mind).}$$

Theorem

$\text{inhab(bool)} = \{\perp, \mathbf{K}, \mathbf{F}, \top\}$.

Proof.

Since bool $<:$ boool, we need only check inhabitants.
All but $\mathbf{J}$ are fixed by make_up_your_mind:

$$\begin{aligned} &(\lambda \text{q. q} \perp (\text{q} \top \perp))\ \mathbf{J} \\ &\quad = \mathbf{J} \perp (\mathbf{J} \top \perp) \\ &\quad = \perp \mid \top \mid \perp \\ &\quad = \top \end{aligned}$$

$\square$

Outline of constructing types

This technique generalizes to more complicated types.

$$
\begin{aligned}
\mathsf{bool} \;:=\; \mathsf{type}\ (&\\
&(\mathsf{Simple}\ \lambda \mathsf{a}, \mathsf{a}'.\ \mathsf{a} \to \mathsf{a} \to \mathsf{a}')\\
&\mid\ \langle \mathbf{K}, \mathbf{F} \rangle\\
&\mid\ \lambda \mathsf{q}.\ \mathsf{q}\ \bot\ (\mathsf{q}\ \top\ \bot)\\
).&
\end{aligned}
$$

(1) enforce simple concurrent typing

(2) descend Böhm tree with intro and elim forms

(3) enforce sequentiality: one head variable only

Product types (actually dropped product)

$$
\begin{aligned}
&\mathsf{Prod} \;:=\; \mathsf{type} \to \mathsf{type} \to \mathsf{type}\ (\\
&\quad \lambda a, b.\ (\mathsf{Simple}\ \lambda c, c'.\ (a \to b \to c) \to c')\\
&\qquad\quad \mid\ \langle \lambda x, y. \langle x, y\rangle\rangle\\
&\qquad\quad \mid\ \lambda q.\ \langle q\ \mathbf{K},\ q\ \mathbf{F}\rangle\\
&).
\end{aligned}
$$

Theorem

For $a, b : \mathsf{type}$, $\mathsf{inhab}(\mathsf{Prod}\ a\ b) = \{\top\} \cup \{\langle x, y\rangle \mid x{:}a, y{:}b\}$

Proof.

Any h.n.f. below $q : \mathsf{Prod}\ a\ b$ must be $\langle a\ x, b\ y\rangle$ or $\top$.
What is the maximal such?
First component is $q\ \mathbf{K}$. Second component is $q\ \mathbf{F}$.
So $\lambda q.\ \langle q\ \mathbf{K},\ q\ \mathbf{F}\rangle$ ensures sequentiality.

$\square$

Sum types (actually dropped, lifted sum)

$$
\begin{array}{l}
\mathsf{Sum} \;:=\; \mathsf{type} \to \mathsf{type} \to \mathsf{type}\ (\\
\quad \lambda a, b.\ (\mathsf{Simple}\ \lambda c, c'.\ (a \to c) \to (b \to c) \to c') \\
\qquad\quad \mid\ \langle \mathsf{inl},\ \mathsf{inr} \rangle \\
\qquad\quad \mid\ \lambda q, f, g.\ q\ (\mathbf{K}\ \mathbf{I})\ \bot\ (q\ f\ \top) \\
\qquad\qquad\qquad\quad \mid\ q\ \bot\ (\mathbf{K}\ \mathbf{I})\ (q\ \top\ g) \\
).
\end{array}
$$

where

$$\mathsf{inl} = \lambda x, f, \_.f\ x, \qquad \mathsf{inr} = \lambda y, \_, g.g\ x$$

Theorem

$\mathsf{inhab}(\mathsf{Sum}\ a\ b) = \{\top, \bot\} \cup \{\mathsf{inl}\ x \mid x{:}a\} \cup \{\mathsf{inr}\ y \mid y{:}b\}.$

Proof.

Combine proofs of bool and Prod.

□

Self-recursing numerals: motivation

Church numerals have simple type $(\mathsf{a} \to \mathsf{a}) \to \mathsf{a} \to \mathsf{a}$.
but predecessor has problems:

- on well-defined terms, it is linear-time.
- on partially-defined terms it diverges.

Gödel's recursor has type $\mathsf{nat} \to (\mathsf{nat} \to \mathsf{a} \to \mathsf{a}) \to \mathsf{a} \to \mathsf{a}$.
For self recursion, redefine $\mathsf{nat} = \mu\ \mathsf{n}.\ (\mathsf{n} \to \mathsf{a} \to \mathsf{a}) \to \mathsf{a} \to \mathsf{a}$:

$$\begin{aligned} \mathsf{zero} &:= \lambda\_, \mathsf{x}.\ \mathsf{x}. \\ \mathsf{succ} &:= \lambda \mathsf{n}, \mathsf{f}, \mathsf{x}.\ \mathsf{n}\ \mathsf{f}\ (\mathsf{f}\ \mathsf{n}\ \mathsf{x}) \end{aligned}$$

These nats are redundant; exponentially large normal forms:

$$2 = \lambda \mathsf{f}, \mathsf{x}.\ \mathsf{f}\ (\lambda \mathsf{f}, \mathsf{x}.\mathsf{f}(\lambda \mathsf{f}, \mathsf{x}.\mathsf{x})\mathsf{x})\ (\mathsf{f}(\lambda \mathsf{f}, \mathsf{x}.\mathsf{x})\mathsf{x})$$

Self-recursing numerals: correctness

$$
\begin{aligned}
&\mathsf{nat} \;:=\; \mathsf{type}\;(\\
&\quad \mathbf{Y}\;\lambda a.\;(\mathsf{Simple}\;\lambda b, b'.\;(a \to b' \to b) \to b \to b') \\
&\qquad \mid\; \langle \lambda n{:}a, \_, f{:}a \to a, x{:}a.\; f\; n(n\; f\; x),\quad \lambda \_, x{:}a.\; x \rangle \\
&\qquad \mid\; \langle \lambda \_, n{:}a, f{:}a \to a, x{:}a.\; f\; n(n\; f\; x),\quad \lambda \_, x{:}a.\; x \rangle \\
&\qquad \mid\; \lambda q.\; q\; \bot\; (q\; \top\; \bot) \\
&).
\end{aligned}
$$

Note the two different ways of descending: left and right.

Theorem

$\mathsf{inhab}(\mathsf{nat}) = \{\top\} \;\cup\; \{\mathsf{succ}^n\; z \mid n \in \mathbb{N},\; z \in \{\bot, \mathsf{zero}\}\}.$

Proof.

As above, only we need to ensure consistency across BT.
At root, descend in either direction.
The a in $(a \to b' \to b) \to b \to b'$ descends below root.

$\square$

Quotient types

What is an r.e. set (of $x : a$'s)?
$\rightarrow$ A sequence : $nat \rightarrow a$? ...but order doesn't matter
$\rightarrow$ A semipredicate : $a \rightarrow semi$? ...but no mapping
$\rightarrow$ A semiset : $(a \rightarrow b) \rightarrow b$? ...works in concurrent CA.

$$\mathsf{Semiset} \;:=\; \mathsf{type} \rightarrow \mathsf{type}\ (\lambda a.\ \mathsf{Simple}\ \lambda b, b'.\ (a \rightarrow b) \rightarrow b').$$

Now we can define quotient types.
Let $M : \mathsf{Semiset}(a \rightarrow a)$ generate a monoid action on a.
The quotient type of M-orbits is Mod M, where

$$\begin{aligned} &\mathsf{Mod} \;:=\; (\forall a : \mathsf{close}.\ (\mathsf{Semiset}\ a \rightarrow a)\ \rightarrow\ (\mathsf{Sub}\ (\mathsf{Semiset}\ a)))\ (\\ &\qquad \lambda a.\ \ \lambda M.M\lambda m.\ \ \lambda X.X\lambda x.\ \ \langle m\ x \rangle \\ &). \end{aligned}$$

Summary and Questions

- Concurrent CA is inadvertently typed
  (sequential CA is not).
- $\mathbf{S}, \mathbf{K}, \mathbf{J}$-definable types required head normal forms:
  $\mathcal{P}\omega$ fails, $\mathcal{D}_\infty$ fails, completed term model works.
- Very rich type structure.

Questions.

- Exactly which types are definable?
- Are sequential simple types uniformly definable?

Definition of raising and lowering operators

Define raising and lowering operators

$$\begin{aligned} \mathsf{raise} &:= (\lambda \mathsf{x}, \_.\mathsf{x}) &= \mathbf{K}. \\ \mathsf{lower} &:= (\lambda \mathsf{x}.\mathsf{x}\ \top) &= \langle \top \rangle. \end{aligned}$$

so that

$$\begin{aligned} &\mathsf{lower} \circ \mathsf{raise} = \mathbf{I}, \\ &\mathsf{raise} \circ \mathsf{lower} = \lambda \mathsf{x}, \_.\ \mathsf{x}\ \top \quad \sqsupseteq \quad \mathbf{I} \end{aligned}$$

Similarly at function type,

$$\begin{aligned} &\mathbf{I} \rightarrow \mathsf{raise} = \lambda \mathsf{f}, \mathsf{x}, \_.\mathsf{f}\ \mathsf{x} \\ &\mathbf{I} \rightarrow \mathsf{lower} = \lambda \mathsf{f}, \mathsf{x}.\ \mathsf{f}\ \mathsf{x}\ \top \end{aligned}$$

so that

$$\begin{aligned} &(\mathbf{I} \rightarrow \mathsf{lower}) \circ (\mathbf{I} \rightarrow \mathsf{raise}) = \mathbf{I}, \\ &(\mathbf{I} \rightarrow \mathsf{raise}) \circ (\mathbf{I} \rightarrow \mathsf{lower}) = \lambda \mathsf{f}, \mathsf{x}, \_.\ \mathsf{f}\ \mathsf{x}\ \top \quad \sqsupseteq \quad \mathbf{I} \end{aligned}$$

Constructing simple concurrent types

Now these operators generate our previous $\mathsf{L}_{mn}, \mathsf{R}_{mn}$:

$$\mathsf{R}_{mn} = (\mathsf{m}\ \mathsf{raise}) \circ (\mathsf{n}\ \mathbf{I} \to \mathsf{raise})$$
$$\mathsf{L}_{mn} = (\mathsf{n}\ \mathbf{I} \to \mathsf{lower}) \circ (\mathsf{m}\ \mathsf{lower})$$

Hence we have a simple definition of semi

$$\begin{array}{l}
\mathsf{Simple} \ := \ \mathsf{any} \to \mathsf{type}\ (\\
\quad \lambda \mathsf{f}.\ \mathsf{curry} \to \mathsf{uncurry} \\
\qquad \mid \ \mathsf{f}\ \mathbf{I}\ \mathbf{I} \\
\qquad \mid \ \mathsf{f}\ \mathsf{raise}\ \mathsf{lower} \\
\qquad \mid \ \mathsf{f}\ \mathbf{I} \to \mathsf{raise}\ \mathbf{I} \to \mathsf{lower} \\
).
\end{array}$$

Applications to typechecking

Now the boolean type becomes

$$\begin{array}{lll} \mathsf{bool} & := & \mathsf{type}\ (\mathsf{curry} \to \mathsf{curry} \to \mathsf{uncurry} \\ & \mid & \mathsf{raise} \to \mathsf{raise} \to \mathsf{lower} \\ & \mid & (\mathbf{I} \to \mathsf{raise}) \to (\mathbf{I} \to \mathsf{raise}) \to (\mathbf{I} \to \mathsf{lower}) \\ & \mid & \langle \mathbf{K}, \mathbf{F} \rangle \\ & \mid & (\lambda \mathsf{q}.\ \mathsf{q} \perp (\mathsf{q} \top \perp)) \\ ). & & \end{array}$$

We can now reduce typechecking $\mathsf{x} : \mathsf{bool}$ to five checks, which may succeed even under β-η conversion!

back

Correctness of semi: overview

$$\mathsf{semi} \;:=\; \mathsf{type}\ ((\mathsf{Simple}\ \lambda a, a'.\ a \to a')\ \mid\ \langle \mathbf{I} \rangle).$$

Theorem

$\mathsf{inhab}(\mathsf{semi}) = \{\bot, \mathbf{I}, \top\}$.

Proof.

$\bot : \mathsf{semi}$ by β-reduction. Any other $q : \mathsf{semi}$ converges, say

$$q \;\sqsupseteq\; q' = \lambda f, x_1, \ldots, x_n.\ z\ M_1\ \ldots\ M_m$$

Show that either $q = \top$ or

$z = f$	*(head is in the right place),*
$m = n$	*(right number of limbs), and*
$M_i \sqsubseteq x_i$	*(each limb is healthy).*

Finally raise a healthy $q' = \lambda f, \underline{x}.f\ \underline{M}$ up to $\mathbf{I}$.

□

Correctness of semi: head is in the right place

We know $\mathsf{q} : \mathsf{semi}$ and

$$\mathsf{q} \;\sqsupseteq\; \mathsf{q}' = \lambda \mathsf{f}, \mathsf{x}_1, \ldots, \mathsf{x}_\mathsf{n}.\ \mathsf{z}\ \mathsf{M}_1\ \ \ldots\ \ \mathsf{M}_\mathsf{m}$$

If $\mathsf{z} \neq \mathsf{f}$ then $\mathsf{z} = \mathsf{x}_\mathsf{i}$ for some i.

Cover all the x_j's with a minefield (n,0):

$$\begin{aligned}
\mathsf{R}\!\rightarrow\!\mathsf{L}\ \mathsf{q}'\ \mathsf{f} &= \ \mathsf{q}\ (\mathsf{n}\ \mathbf{K}\ \mathsf{f})\ \top^{\sim\mathsf{n}} \\
&= \ (\lambda \underline{\mathsf{x}}.\ \mathsf{x}_\mathsf{i}\ \underline{\mathsf{M}})\ \top^{\sim\mathsf{n}} \\
&= \ \top\ \underline{\mathsf{M}} \qquad = \ \top
\end{aligned}$$

So $\mathsf{q} = \top$.
otherwise...

Correctness of semi: right number of limbs

We know q : semi and

$$q \sqsupseteq q' = \lambda f, x_1, \ldots, x_n.\ f\ M_1\ \ \ldots\ \ M_m$$

Make q' navigate a big minefield, say $(n + m, n + m)$

$$\begin{aligned} R \to L\ q'\ f\ x = &\ \ q'\ (\lambda \underline{u}, v, \underline{w}.\ f\ v)\ T^{\sim m+n}\ x\ T^{\sim m+n} \\ = &\ \ (\lambda \underline{x}, \underline{u}, v, \underline{w}.\ f\ v)\ \underline{M}\ \top^{\sim m+n}\ x\ \top^{\sim m+n} \end{aligned}$$

How far off can q' be?

$$|\ \underline{x}, \underline{u}\ | = 2n + m \quad \stackrel{?}{=} \quad n + 2m = |\ \underline{M}\ \top^{\sim m+n}\ \ |$$

If $n \neq m$ then semi $q' = \top$.
otherwise...

Correctness of semi: each limb is healthy

We know q : semi and

$$q \;\sqsupseteq\; q' = \lambda f, x_1, \ldots, x_n.\ f\ M_1\ \ldots\ M_n$$

If $M_i \not\sqsupseteq x_i$ then $M_i \sqsupseteq N \not\sqsupseteq x_i$ for some h.n.f N.
Somewhere down the BT of q' is either a $\top$,
or an offending head variable z.
If $z \notin \{f, \underline{x}\}$, descend with $\langle \mathbf{I} \rangle$ until it is.
If $z = x_i$, make q navigate a minefield; then descend.
If $z = f$, make f inert with curry; then descend.

Eventually we hit a $\top$.
otherwise...

Correctness of semi: raising partial terms up to $\mathbf{I}$

We know q' is healthy, but not at full strength

$$\lambda f, \underline{x}.f\ \underline{x} \quad \sqsupseteq \quad q' \quad \sqsupseteq \quad \lambda f, \underline{x}.\ f\ \bot^{\sim n}$$

Raise and lower n times to ignore faulty args

$$\begin{aligned} & n\ \mathbf{K} \rightarrow \langle \top \rangle\ q'\ f \\ & \quad = \ q'\ (n\ \mathbf{K}\ f)\ \top^{\sim n} \\ & \quad = \ (\lambda \underline{x}.\ n\ \mathbf{K}\ f\ \underline{M})\ \top^{\sim n} \\ & \quad = \ (\lambda \underline{x}.\ f)\ \top^{\sim n} \\ & \quad = \ f \end{aligned}$$

So finally $q \sqsupseteq \text{semi}\ q' = \mathbf{I}$. back