ADVENTURES IN TIME & SPACE

Jim Royer
Syracuse University

Joint work with
Norman Danner
Wesleyan University

GeoCal'06

Based on: Adventures in Time and Space, by N. Danner and J.S. Royer, Proceedings of the 2006 ACM Principles of Programming Languages Conference (POPL), 2006.

THE PROBLEM

Solve for X in

$$(\text{PCF} - \text{fix}) + X = \begin{cases} \text{a serviceable programming language} \\ \text{for higher-type polynomial time} \end{cases}$$

where

- X's constraints are via typing (implicit complexity)
- serviceable $\approx$ lots of algorithms are directly expressible
- X includes something "close to" fix

PCF =

simply-typed λ-calc. + basic string ops + fixed pt. comb. (call-by-value)

higher-type poly-time =

Mehlhorn's and Cook-Urquhart's basic feasible functionals (more later)

WHY THIS PROBLEM?

It is at a crossroads of many interesting paths.

- **Efficiency as a safety property**
  E.g., for proof-carrying code
- **Street-level view of higher types**
  E.g., Bird's chapter on efficiency:
  - Cunning, sound program transforms,
  - These can be used achieve efficiency
  - But how to do so? . . . only heuristic advice and (wonderful) examples.
- **Applications in Cryptography and Learning**
  E.g., Transformations on pseudo-random generators
- . . .
- **Insight into higher-type complexity**

TWO APPROACHES TO HIGHER TYPE POLY-TIME

Higher types over ordinary polynomial time

Focus: Bringing higher types & other modern accoutrements into the programming type-1 poly-time functions.

Hard Part: Gracefully handling the inevitable restrictions.

Examples: Aehlig, Bellantoni, Hofmann, Niggl, Schwichtenberg, ...

Polynomial time over higher type objects

Focus: Bringing complexity theoretic concerns and questions into the realm of higher types. **(Why bother?)**

Hard Part: Re-thinking the world.

E.g., What is the "computational complexity" of $(f, g) \mapsto f \circ g$?

Focus of on type-level 2 (Type-levels 3, 4, 5, ...? another day)

Examples: Kapron-Cook, Seth, ..., this talk

TYPE-2 POLY-TIME I: SIZES

Conventions

- $\mathbb{N} =_{\text{def}} \{0,1\}^* \equiv$ **values**, to be computed over
- $\omega =_{\text{def}} \{0\}^* \equiv$ **tallies**, results of size measurements

Measuring sizes

- $|x| =_{\text{def}} 0^{\ell}$, where $\ell =$ length of $x \in \mathbb{N}$. E.g., $|101| = 000$.
- For $f\colon \mathbb{N} \longrightarrow \mathbb{N}$, we have $|f|\colon \omega \longrightarrow \omega \ni$ for each $n \in \omega$,

  $$|f|(n) =_{\text{def}} \max(\{\, |f(x)| \;:\; |x| \leq n \,\}).$$ (Kapron-Cook)

  E.g., $|\lambda x \in \mathbb{N}\,.\,(x \oplus x)| = \lambda n \in \omega\,.\,2n$.
- and so on for type-2 functions, types, type-contexts, ...

$|\cdot|$: the realm of values $\xrightarrow{\text{“functorially”}}$ the realm of sizes

TYPE-2 POLY-TIME II: SECOND-ORDER POLYNOMIALS

Example For $C = \lambda f, g \,.\, \lambda x \,.\, f(g(x))$:

$$|C(f,g)(x)| \leq \underbrace{|f|(|g|(|x|))}_{\text{poly over } |f|,|g|,|x|} \,.$$

The second-order polynomials: Syntax (Kapron-Cook)

(the simply-typed λ-calculus over base type T)$|_{\text{type levels} \leq 2}$

$+$

tally constants and type-1 binary ops $+$, $*$, and $\vee$

The second-order polynomials: Semantics ($\mathcal{L}[\![\,\cdot\,]\!]$)

$\mathcal{L}[\![\mathsf{T}]\!] =_{\text{def}} \omega.$ $\quad \mathcal{L}[\![\mathsf{T} \to \mathsf{T}]\!] =_{\text{def}} \omega \Rightarrow \omega.$ $\quad \ldots$

$\mathcal{L}[\![x \vee y]\!] =_{\text{def}} \max(x,y).$ $\quad \mathcal{L}[\![x + y]\!] =_{\text{def}} x + y.$ $\quad \ldots$

TYPE-2 POLY-TIME III: THE BFFS

THE KAPRON-COOK THEOREM

$F: (\mathbb{N} \longrightarrow \mathbb{N}) \times \mathbb{N} \longrightarrow \mathbb{N}$ is a basic feasible functional iff

there is a machine M and

there is second-order poly $q \ni$

$\forall f, x \quad M$ on input (f, x)

(a) outputs $F(f, x)$, and

(b) runs within time $q(|f|, |x|)$.

Machines & costs: anything "sensible" works

We use a CEK-abstract machine for PCF with a particular cost measure.

Now back to our problem

INGREDIENTS OF X: TIERED BASE TYPES

(PCF − fix) + X = a nice PL for type-level 2 poly-time

Labels:

$$\varepsilon < \diamond < \Box\diamond < \diamond\Box\diamond < \Box\diamond\Box\diamond < \diamond\Box\diamond\Box\diamond < \ldots$$

Base Types:

$$\mathbf{N}_\varepsilon \leq: \mathbf{N}_\diamond \leq: \mathbf{N}_{\Box\diamond} \leq: \mathbf{N}_{\diamond\Box\diamond} \leq: \mathbf{N}_{\Box\diamond\Box\diamond} \leq: \ldots$$

Intuitive Interpretation: The labels describe size bounds

$$\begin{aligned}
x{:}\,\mathbf{N}_\varepsilon &\approx |x| \leq |\text{some input string}| \\
x{:}\,\mathbf{N}_\diamond &\approx |x| \leq \text{poly}(|\text{some input string}|) \\
x{:}\,\mathbf{N}_{\Box\diamond} &\approx |x| \leq |\text{f}|(\text{poly}(|\text{some input string}|)) \\
x{:}\,\mathbf{N}_{\diamond\Box\diamond} &\approx |x| \leq \text{poly}(|\text{f}|(\text{poly}(|\text{some input string}|))) \\
x{:}\,\mathbf{N}_{\Box\diamond\Box\diamond} &\approx |x| \leq |\text{f}|(\text{poly}(|\text{f}|(\text{poly}(|\text{some input string}|)))) \\
&\ldots
\end{aligned}$$

(Bellantoni & Cook connection: $\mathbf{N}_\varepsilon \approx$ **normal** $\mathbf{N}_\diamond \approx$ **safe)**

INGREDIENTS OF X: ARROW TYPES

(PCF − fix) + X = a nice PL for type-level 2 poly-time

Arrow types = the simple types over the base types

E.g., $f\colon \mathrm{N}_{\diamond} \to \mathrm{N}_{\Box\diamond}$

But how to type $f(f(x))$**?**

Subsumption + covariant shifting of arrow types

E.g., $f\colon \mathrm{N}_{\diamond} \to \mathrm{N}_{\Box\diamond} \quad \Longrightarrow \quad f\colon \mathrm{N}_{\diamond\Box\diamond} \to \mathrm{N}_{\Box\diamond\Box\diamond}$

But, we need to control more than size . . .

INGREDIENTS OF X: CREC & IMPLICIT $\multimap$-TYPES

(PCF − fix) + X = a nice PL for type-level 2 poly-time

Clocked Recursion: crec $a\ (\lambda_r f\,.\,E)$

For constructive runtime bounds on programs,
something equivalent to clocking is necessary.

One use recursion: f as above is affinely restricted

This allows us to handle "linear," poly-depth recursions.
Provides: $\mathbf{Time}(m, \vec{n}) \leq \mathbf{Time}(m-1, \vec{n}) + \mathbf{poly}(\vec{n})$
Built on Plotkin and Barber's DILL (E.g., $\Gamma; f\!:\!\sigma \vdash E\!:\!\sigma \to \sigma$)

Tail recursion:

Mainly for simplicity.
Nearly everyone else uses primitive recursions.

Putting things together . . .

ATR: AFFINE TAIL-RECURSION

Grammar of Raw Expressions

$$E ::= K \mid (\mathsf{c_a}\ E) \mid (\mathsf{d}\ E) \mid (\mathsf{t_a}\ E) \mid (\mathsf{down}\ E\ E) \mid V \mid (E\ E) \mid (\lambda V\,.\,E) \mid (\mathsf{if}\ E\ \mathsf{then}\ E\ \mathsf{else}\ E) \mid (\mathsf{crec}\ K\ (\lambda_r V\,.\,E))$$

$$K ::= \{0,1\}^*$$

Rewrite Rules:

($\oplus$ = string concatenation)

$(\mathsf{c_a}\ x) \rightsquigarrow \mathsf{a} \oplus x.$ $\quad (\mathsf{d}\ (\mathsf{a} \oplus x)) \rightsquigarrow x.$ $\quad (\mathsf{d}\ \epsilon) \rightsquigarrow \epsilon.$

$$(\mathsf{t_a}\ x) \rightsquigarrow \begin{cases} 0, & \text{if } x \text{ begins with a;} \\ \epsilon, & \text{otherwise.} \end{cases}$$

$$(\mathsf{down}\ x\ y) \rightsquigarrow \begin{cases} x, & \text{if } |x| \leq |y|; \\ \epsilon, & \text{otherwise.} \end{cases}$$

$$(\mathsf{if}\ x\ \mathsf{then}\ y\ \mathsf{else}\ z) \rightsquigarrow \begin{cases} y, & \text{if } x \neq \epsilon; \\ z, & \text{if } x = \epsilon. \end{cases}$$

more..

ATR: MORE REWRITE RULES

Call-By-Value β-Reduction

As usual

A Standard Call-By-Value Rewrite Rule for fix (Not part of ATR!)

$$\textsf{fix}\ (\lambda f\,.\,E) \rightsquigarrow E[f := (\textsf{fix}\ (\lambda f\,.\,E)].$$

The Rewrite Rule for crec

$$\textsf{crec}\ a\ (\lambda_r f\,.\,E) \rightsquigarrow \lambda \vec{v}\,.\ (\textsf{if}\ |a| \leq |v_1|\ \textsf{then}\ (E'\ \vec{v}\,)\ \textsf{else}\ \epsilon)$$
$$\textsf{with}\ E' = E[f := \Big(\textsf{crec}\ \ (0 \oplus a)\ (\lambda_r f\,.\,E)\Big)],$$

- $a \approx$ the internal clock — that counts up.
- $0 \oplus a \approx$ a tick of the clock
- Typing constraints will make sure $|v_1|$ is bounded.

ATR: TYPING I

(Zero-I) $$\frac{}{\Gamma; \Delta \vdash \epsilon : \mathrm{N}_{\varepsilon}}$$

(Const-I) $$\frac{}{\Gamma; \Delta \vdash K : \mathrm{N}_{\diamond}}$$

(Int-Id-I) $$\frac{}{\Gamma,\ v : \sigma; \Delta \vdash v : \sigma}$$

(Aff-Id-I) $$\frac{}{\Gamma; v : \gamma \vdash v : \gamma}$$

(Subsumption) $$\frac{\Gamma; \Delta \vdash E : \sigma}{\Gamma; \Delta \vdash E : \tau}\ (\sigma \leq : \tau)$$

(op-I) $$\frac{\Gamma; \Delta \vdash E : \mathrm{N}_{\diamond_d}}{\Gamma; \Delta \vdash (\mathbf{op}\ E) : \mathrm{N}_{\diamond_d}}$$

op ranges over c_0, c_1, d, t_0, and t_1

(Shift) $$\frac{\Gamma; \Delta \vdash E : \sigma}{\Gamma; \Delta \vdash E : \tau}\ (\sigma \propto \tau)$$

(down-I) $$\frac{\Gamma; \Delta \vdash E : \mathrm{N}_L \qquad \Gamma; \Delta' \vdash E' : \mathrm{N}_{L'}}{\Gamma; \Delta, \Delta' \vdash (\mathbf{down}\ E\ E') : \mathrm{N}_{L'}}$$

(if-I) $$\frac{\Gamma; \_ \vdash E_0 : \mathrm{N}_L \qquad \Gamma; \Delta_1 \vdash E_1 : \mathrm{N}_{L'} \qquad \Gamma; \Delta_2 \vdash E_2 : \mathrm{N}_{L'}}{\Gamma; \Delta_1 \cup \Delta_2 \vdash (\mathbf{if}\ E_0\ \mathbf{then}\ E_1\ \mathbf{else}\ E_2) : \mathrm{N}_{L'}}$$

ATR: TYPING II

$$(\to\text{-I})\quad \frac{\Gamma,\, v{:}\,\sigma;\Delta \vdash E{:}\,\tau}{\Gamma;\Delta \vdash (\lambda v\,.\,E){:}\,\sigma \to \tau}$$

$$(\to\text{-E})\quad \frac{\Gamma;\Delta \vdash E_0{:}\,\sigma \to \tau \qquad \Gamma;\_ \vdash E_1{:}\,\sigma}{\Gamma;\Delta \vdash (E_0\ E_1){:}\,\tau}$$

$$(\textbf{crec–I})\quad \frac{\vdash K{:}\,\mathrm{N}_\diamond \qquad \Gamma;f{:}\,\gamma \vdash E{:}\,\gamma}{\Gamma;\_ \vdash (\textbf{crec}\ K\ (\lambda_r f\,.\,E)){:}\,\gamma}\quad \Big(\textbf{TailPos}(f,E)\ \textbf{and}\ \gamma \in \mathcal{R}\Big)$$

where:

$$\textbf{TailPos}(f,E) \ =_{\mathrm{def}}\ \left[\begin{array}{l}\textbf{Each occurrence of } f \textbf{ in } E\\ \textbf{is as the head of a tail call}\end{array}\right].$$

$$\mathcal{R} \ =_{\mathrm{def}}\ \{\,(\mathrm{b}_1,\mathrm{b}_2,\ldots,\mathrm{b}_k)\to \mathrm{b} \ \mid\ \mathrm{b}_1 \textbf{ and each } \mathrm{b}_i \leq{:}\, \mathrm{b}_1 \textbf{ is oracular}\,\}\,.$$

The oracular base types $= \mathrm{N}_\varepsilon,\ \mathrm{N}_{\square\diamond},\ \mathrm{N}_{\square\diamond\square\diamond},\ \mathrm{N}_{\square\diamond\square\diamond\square\diamond},\ \ldots$

EXAMPLE: REVERSE

$$
\begin{array}{l}
\textsf{reverse} : \mathrm{N}_\varepsilon \to \mathrm{N}_\diamond \;=\\
\quad \lambda w\,.\ \textsf{letrec}\ f : \mathrm{N}_\varepsilon \to \mathrm{N}_\diamond \to \mathrm{N}_\diamond \to \mathrm{N}_\diamond \;=\\
\qquad \lambda b,\, x,\, r\,.\, \textsf{if}\ (\mathrm{t}_0\ x) \qquad \textsf{then}\ f\ b\ (\mathrm{d}\ x)\ (\mathrm{c}_0\ r)\\
\qquad\qquad\qquad\quad \textsf{else if}\ (\mathrm{t}_1\ x)\ \textsf{then}\ f\ b\ (\mathrm{d}\ x)\ (\mathrm{c}_1\ r)\\
\qquad\qquad\qquad\quad \textsf{else}\ r\\
\qquad\quad \textsf{in}\ f\ w\ w\ \epsilon
\end{array}
$$

- $(\textsf{letrec}\ \ f = D\ \ \textsf{in}\ \ E)\ \equiv\ E[f := (\textsf{crec}\ \epsilon\ (\lambda_r f\,.\,D))]$
- Recall:
  - $(\mathrm{t_a}\ x) \equiv [x\ \textsf{starts with a?}]$.
  - $(\mathrm{c_a}\ x) = \mathrm{a} \oplus x$.
  - $(\mathrm{d}\ (\mathrm{a} \oplus x)) = x$.
- b - programmer's bound on the number of recursions

EXAMPLE: PRIM. REC. ON NOTATION

$$
\begin{array}{l}
\mathrm{prn}\colon (\mathrm{N}_\diamond \to \mathrm{N}_\diamond \to \mathrm{N}_\diamond) \to \mathrm{N}_\varepsilon \to \mathrm{N}_\diamond \;=\\
\quad \lambda g,\, y\,.\ \textsf{letrec}\ f\colon \mathrm{N}_\varepsilon \to \mathrm{N}_\diamond \to \mathrm{N}_\diamond \to \mathrm{N}_\diamond \to \mathrm{N}_\diamond \;=\\
\qquad\quad \lambda b,\, x,\, z,\, r\,.\\
\qquad\qquad \textsf{if}\ (\mathrm{t}_0\ x) \quad\ \textsf{then}\ f\ b\ (\mathrm{d}\ x)\ (\mathrm{c}_0\ z)\ (g\ (\mathrm{c}_0\ z)\ r)\\
\qquad\qquad \textsf{else if}\ (\mathrm{t}_1\ x)\ \textsf{then}\ f\ b\ (\mathrm{d}\ x)\ (\mathrm{c}_1\ z)\ (g\ (\mathrm{c}_1\ z)\ r)\\
\qquad\qquad \textsf{else} \qquad\qquad\qquad\ r\\
\qquad\ \textsf{in}\ f\ y\ (\textsf{reverse}\ y)\ \epsilon\ (g\ \epsilon\ \epsilon)
\end{array}
$$

where

- $$\left\{\begin{array}{rcl} \mathrm{prn}\ g\ \epsilon & \rightsquigarrow & g\ \epsilon\ \epsilon. \\ \mathrm{prn}\ g\ (\mathrm{a} \oplus y) & \rightsquigarrow & g\ (\mathrm{a} \oplus y)\ (\mathrm{prn}\ g\ y). \end{array}\right\} \qquad (*)$$
- As before

  $(\textsf{letrec}\ \ f = D\ \ \textsf{in}\ \ E)\ \equiv\ E[f := (\textsf{crec}\ \epsilon\ (\lambda_r f\,.\,D))]$ and

  b - programmer's bound on the number of recursions

N.B. The prn functional as defined by $(*)$, **is not** a BFF.
The side-conditions that tame $(*)$ are part of **ATR**'s semantics.

EXAMPLE: KAPRON'S FUNCTIONAL

```
kfun: (N◇ → N□◇) → Nε → Nε  =    // Computes K given below
    λf, x .  letrec h: N□◇ → Nε → Nε =
                  λm, k .  // Invariant: k ≤ len(m) and |m| ≤ |f|(|x|)
                     if (k == x) or (k == (len m))
                       then k
                       else h (max (f (k + 1)) m) (down (k + 1) x)
              in  h (f ϵ) ϵ
```

(Fixed from the POPL proceedings.)

where

- $\mathrm{len}(z)$ = the dyadic representation of the length of z.
- $K(f,x) = \begin{cases} (\mu k < x)\left[k = \max_{i \le k} \mathrm{len}(f(i))\right], & \text{if such a } k \text{ exists;} \\ x, & \text{otherwise;} \end{cases}$
- various secondary functions are given (correctly typed) definitions

N.B. K is a BFF and the key example that lead to the Kapron-Cook Thm.

POLYNOMIAL-SIZE BOUNDEDNESS FOR ATR

THEOREM

If $\Gamma; \Delta \vdash E : \sigma$,

then there is a $|\Gamma; \Delta| \vdash q_E : |\sigma| \ni$

$$|\mathcal{V}_{\mathrm{wt}}[\![E]\!]\,\rho| \quad \leq \quad \mathcal{L}_{\mathrm{wt}}[\![q_E]\!]\,|\rho|, \text{ for all } \rho \in \mathcal{V}_{\mathrm{wt}}[\![\Gamma; \Delta]\!].$$

- 2nd-order polys are also typed. E.g., $|f| : \mathsf{T}_{\diamond} \longrightarrow \mathsf{T}_{\square\diamond}$
- $\mathcal{V}_{\mathrm{wt}}[\![\cdot]\!]$ = the naive value semantics, pruned.

  where the naive semantics $\mathcal{V}[\![\cdot]\!]$ = a standard Scott semantics for PCF.
- $\mathcal{L}_{\mathrm{wt}}[\![\cdot]\!]$ = the naive length semantics, pruned.

  where the naive semantics $\mathcal{L}[\![\cdot]\!]$ = as before.
- This pruning the subtlest part of the paper.

a few details . . .

PREDICATIVE/IMPREDICATIVE AND FLAT/STRICT

Definitions

- $\mathbf{tail}(\mathrm{N}_L) = \mathrm{N}_L$. $\quad\quad$ $\mathbf{tail}(\sigma \to \tau) = \mathbf{tail}(\tau)$.
- γ is predicative when γ is a base type or else $\gamma = \sigma \to \tau$ where τ is predicative and $\mathbf{tail}(\sigma) \leq: \mathbf{tail}(\tau)$.
- $\sigma_1 \to \cdots \to \sigma_k \to \mathrm{N}_L$ is flat when $(\exists i)\ \mathbf{tail}(\sigma_i) = \mathrm{N}_L$.
- impredicative $\equiv \neg$predicative $\quad\quad$ - strict $\equiv \neg$flat

Examples

	predicative	impredicative
strict	$\mathrm{N}_\varepsilon \to \mathrm{N}_\diamond$	$\mathrm{N}_\diamond \to \mathrm{N}_\varepsilon$
flat	$\mathrm{N}_\diamond \to \mathrm{N}_\diamond$	$\mathrm{N}_\diamond \to \mathrm{N}_{\Box\diamond} \to \mathrm{N}_\diamond$

. . . and the point is?

The semantic interpretations of impredicative and flat types must be restricted. Why?

TWO COUNTER-EXAMPLES

$E_1 : \mathrm{N}_\varepsilon \to \mathrm{N}_\diamond \;=$ // Assume $g_1 : \mathrm{N}_\diamond \to \mathrm{N}_\varepsilon$.

λw . let $h_1 : \mathrm{N}_\diamond \to \mathrm{N}_\diamond \to \mathrm{N}_\diamond =$

$\lambda x, y$. if $x \neq \epsilon$ then $(dup\ (g_1\ y)\ (g_1\ y))$ else w

in $\mathrm{prn}\ h_1\ w$

$E_2 : \mathrm{N}_\varepsilon \to \mathrm{N}_\diamond \;=$ // Assume $g_2 : \mathrm{N}_\diamond \to \mathrm{N}_\diamond$.

λw . let $h_2 : \mathrm{N}_\diamond \to \mathrm{N}_\diamond \to \mathrm{N}_\diamond =$

$\lambda x, y$. if $x \neq \epsilon$ then $(g_2\ y)$ else w

in $\mathrm{prn}\ h_2\ w$

where

- (let $x = D$ in E) $\equiv E[x := D]$
- $(\mathrm{prn}\ f\ (a \oplus y)) \rightsquigarrow (f\ (a \oplus y)\ (\mathrm{prn}\ f\ y))$
- $|(dup\ x\ y)| = |x| \cdot |y|$
- $(\mathrm{prn}\ f\ \epsilon) \rightsquigarrow (f\ \epsilon\ \epsilon)$

NOTE:

- If $\rho_1(g_1) = \lambda z \in \mathbb{N} \,.\, z$,
  then $|\mathcal{V}[\![E_1]\!]\,\rho_1| = \lambda n \in \omega \,.\, n^{2^n}$. A normality violation.
- If $\rho_2(g_2) = \lambda z \in \mathbb{N} \,.\, z \oplus z$,
  then $|\mathcal{V}[\![E_2]\!]\,\rho_2| = \lambda n \in \omega \,.\, n \cdot 2^n$. A safety violation.

PRUNING THE NAIVE SEMANTICS

For the impredicative types:

- Note:
  - $|(\textbf{down } x\ y)| \leq |y|$
  - $|(\textbf{if } x \textbf{ then } y \textbf{ else } z)| \leq |y| \vee |z|$
- So prune the semantics to make sure that all values have "strictly predicative upper bounds."

For flat types:

- Recall Bellantoni & Cook's poly-max bounds

$$|f(\vec{x};\vec{y})| \leq p(\,|\overrightarrow{x}|\,) + \bigvee |\overrightarrow{y}|$$

- Flat types have to be bounded by higher-type analogues of the poly-max bounds.
- This is hard work and not for today.

BOUNDING TIME

- $|\mathcal{V}_{\mathrm{wt}}[\![E]\!]\,\rho|$ is an extensional property of E.
- Whereas the time needed to evaluate E is certainly not.

$\therefore$ We introduce $\mathcal{T}[\![E]\!] =$ time complexity of E.

- With size, the $|\cdot|$ "functor" dictates the shape of things.
- What happens with time? Some examples:

TIME COMPLEXITIES I: CURRYING AND TIME

Case 1: $E\colon \mathrm{N}_\varepsilon$

$t \in \omega \quad \ni \quad t =$ the time required to compute E's value

Think of this as time passed

Case 2: $E\colon \mathrm{N}_\varepsilon \to \mathrm{N}_\diamond$

$t\colon \omega \to \omega \quad \ni \quad t(n) =$ the time to evaluate E on an size-n input.

Think of this as time in possible futures.

Case 3: $(E_0\ E_1)\colon \mathrm{N}_\varepsilon \to \mathrm{N}_\diamond$, where $E_0\colon \mathrm{N}_\varepsilon \to \mathrm{N}_\varepsilon \to \mathrm{N}_\diamond$ and $E_1\colon \mathrm{N}_\varepsilon$

$(E_0\ E_1)$ has "time complexities" in the senses of both Cases 1 and 2.

So $(E_0\ E_1)$ has a past and futures of interest.

Case 4: E_0 as above.

E_0 has "time complexities" in the senses of Cases 1, 2, & 3.

Case 5: Consider how type-2 terms complicate things.

Is it all clear now?

TIME COMPLEXITIES II: THE BASIC SCHEME

- A time complexity for E has two components, a cost and a potential.
- The cost (a positive elm. of ω) $\approx$ the cost to evaluate E.
- The form of the potential depends on E's type.
  - For E of base type, the potential $= \max(1, |\text{the value of } E|)$. (Why?)
  - For $E: \mathrm{N}_\varepsilon \to \mathrm{N}_\diamond$ we take the potential to be $p_E: \omega^2 \to \omega^2$ $\ni$
    $p_E(c_{\text{arg}}, p_{\text{arg}}) = (c_{\text{res}}, p_{\text{res}})$ where
    - $(c_{\text{arg}}, p_{\text{arg}})$ = the t.c. of an argument, A, to E
    - $(c_{\text{res}}, p_{\text{res}})$ = the t.c. of $(E\ A)$'s result
  - For E of type $\sigma \to \tau$ our motto is:

    a potential for a thing of type $\sigma \to \tau$ is a map
    from time complexities for things of type σ
    to time complexities for things of type τ.

Others have used the cost/pot. decomposition: Sands, Shultis, van Stone

TIME COMPLEXITIES III: TYPE INTERPRETATION

A type translation For ATR types N_L, σ, τ:

$$\begin{array}{rcll} \|\sigma\| & =_{\mathrm{def}} & \mathsf{T} \times \langle\!\langle \sigma \rangle\!\rangle & \text{the type of a t.c. of a type } \sigma \text{ thing} \\ \langle\!\langle \mathrm{N}_L \rangle\!\rangle & =_{\mathrm{def}} & \mathsf{T}_L & \text{the type of a pot. of a type } \mathrm{N}_L \text{ thing} \\ \langle\!\langle \sigma \to \tau \rangle\!\rangle & =_{\mathrm{def}} & \|\sigma\| \to \|\tau\| & \text{the type of a pot. of a type } \sigma \to \tau \text{ thing} \end{array}$$

DEFINITION (The $\mathcal{T}[\![\cdot]\!]$-interpretation of ATR types)

$$\mathcal{T}[\![\sigma]\!] =_{\mathrm{def}} \mathcal{L}_{\mathrm{wt}}[\![\|\sigma\|]\!].$$

Do interesting, sensible things live in these spaces?

CONNECTING $\mathcal{T}[\![E]\!]$ TO ATR

1. We use our machine cost-model and the $\mathcal{T}[\![\sigma]\!]$ spaces to build a model for the simply typed λ-calculus.

   E.g., $\mathcal{T}$-application. Suppose:

   $(c_{\text{opr}}, p_{\text{opr}}) \in \mathcal{T}[\![\sigma \to \tau]\!]$

   $(c_{\text{arg}}, p_{\text{arg}}) \in \mathcal{T}[\![\sigma]\!]$

   $(c_{\text{res}}, p_{\text{res}}) \in \mathcal{T}[\![\tau]\!]$, where $(c_{\text{res}}, p_{\text{res}}) = p_{\text{opr}}\,(\,(c_{\text{arg}}, p_{\text{arg}})\,)$

   Then $(c_{\text{opr}}, p_{\text{opr}}) \star (c_{\text{arg}}, p_{\text{arg}}) =_{\text{def}} (c_{\text{opr}} + c_{\text{arg}} + c_{\text{res}} + 3,\ p_{\text{res}})$.

   Similarly, for currying, environments, ...

   Note: Things are quite intensional here. E.g., The η-law fails.

2. Based on this model we define $\mathcal{T}[\![E]\!]$ for ATR$^-$ (= ATR sans crec) and show soundness & polytime boundedness. (Details shortly)

3. Then the affine types earn their keep ...

AFFINE DECOMPOSITION

DEFINITION

$$(c_0, p_0) \uplus (c_1, p_1) =_{\text{def}} (c_0 + c_1, p_0 \vee p_1)$$

THEOREM

Suppose $\Gamma; f\!:\gamma \vdash E\!:\mathrm{b}$. Then $\forall\ \varrho \in \mathcal{T}[\![\Gamma; f\!:\gamma]\!]$:

$$\mathcal{T}[\![E]\!]\varrho \ \leq\ \mathcal{T}[\![\underbrace{E[f := \lambda\vec{x}\,.\,\epsilon]}_{f \text{ trivialized}}]\!]\,\varrho \ \uplus\ (\mathcal{T}[\![f]\!] \star \vec{t}\,)\,\varrho$$

where the $\vec{t}$'s are max's over appropriate subterms of E.

This decomposition sets us up for solving the recurrences in . . .

POLYNOMIAL-TIME BOUNDEDNESS

THEOREM (Soundness and Polynomial Boundedness ATR)

(a) The $\mathcal{T}[\![\cdot]\!]$ is sound for ATR wrt the costs for our machine model.

(b) Given $\Gamma; \Delta \vdash_{\mathsf{ATR}} E : \gamma$, $\exists_{\text{effectively}}\ q_E$, a poly. time bound for E.

- polynomial time-boundedness $\approx \mathcal{T}[\![E]\!]$ is not too big
- soundness $\approx \mathcal{T}[\![E]\!]$ is not too small
  $\approx \mathcal{T}[\![E]\!]$ gives an upper bound on the cost of E evaluation
- The poly-size boundedness results play a central role here.

Also

THEOREM (Computational Completeness)

All of the type-2 BFFs are ATR-computable.

ATR: SUMMARY

- **ATR** is far from perfect: [audience inserts long list here]
- **ATR** is a demonstration of concept
  - **ATR** and its complexity properties are treated via standard PL tools
  - $\mathcal{V}_{\mathrm{wt}}[\![\cdot]\!]$, a denotational semantics for a ramified type system
  - $\mathcal{T}[\![\cdot]\!]$, a (machine dependent) semantics of time complexity.
  - These semantics reveal complexity theoretic details.
  - The type system, while ad hoc, has a sound intuitive story behind it.
    $\therefore$ There is some hope programmers could think in it.

POSSIBLE EXTENSIONS

- Higher-type parameters in recursions (continuations)
- other recursion patterns (prim. rec., tree rec., etc.)
- Call-by-name/call-by-need
- Other machine+cost choices
- lists, trees, streams, ...
- type-level 3 and beyond
- making the type system less ad hoc
- ...

Next goal:

"1^{st} order" call-by-(name/need) + streams of strings

Near term goal:

Go through Bird, Paulson, ...and by annotating types and minor reprogramming show many/most programs are poly-time.

FINAL THOUGHT

A Lisp programmer knows the value of everything, but the cost of nothing. — Alan Perlis

Let us work on removing every excuse
a programmer might have for this.