Automated Complexity Analysis of Nuprl Extracts

Ralph Benzinger
Department of Computer Science Δ Cornell University
ralph@cs.cornell.edu

Nuprl Seminar
31 January 2000

Automated Complexity Analysis

Not only correct programs, but efficient programs
Requires calculus to express computational complexity

Complexity of Nuprl Extracts

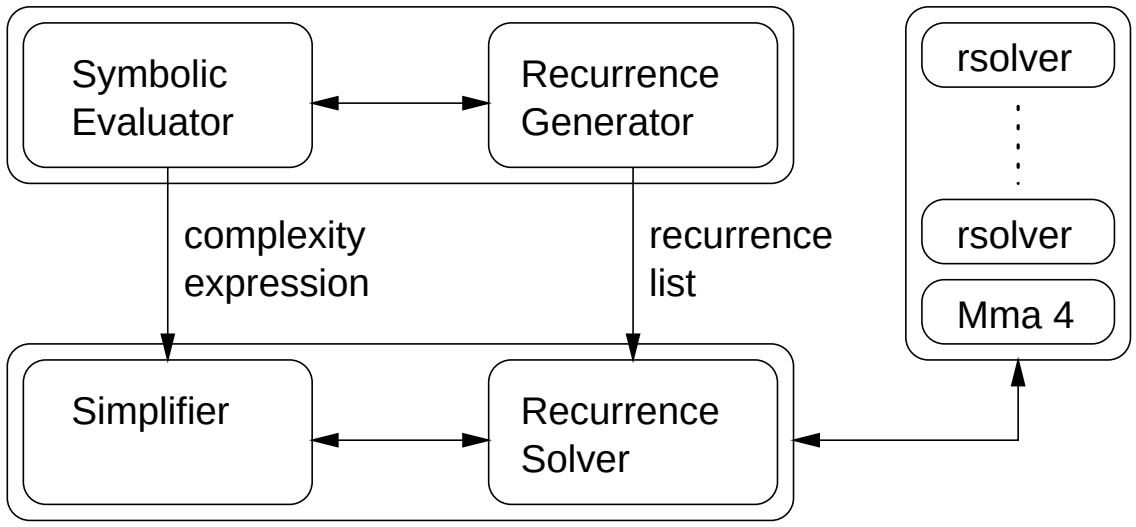
Takes great experience to control running time of extract
("Pigeon-Hole Incident")

Two approaches:
1. Restrict logic to force particular behavior
   $\forall x.\exists_{poly} y.P(x,y)$
2. Construct conventional proof and assert behavior
   $\forall x : Term.\exists f : \mathbb{N} \to Term.\exists n : \mathbb{N}.n = poly(|x|)\&canon(f(n))\& \ldots$

We follow second approach:
ACA Prototype derives upper bound on worst-case time complexity of functional programs written in Nuprl's term language

The ACA*p* System

Symbolic evaluator: abstract interpretation of programs
Recurrence generator: description of inductive terms
Recurrence solver: modular, uses pattern matching
Simplifier: term simplification and approximation heuristics
Rsolvers: handle one particular class of r.e. each



3000 lines of ML and 300 lines of Mathematica code

Previous Work

Systems

Wegbreit (1975): METRIC – first-order LISP, call-by-value
requires known probability distribution for conditional tests

LeMétayer (1985): ACE – FP programs, call-by-value
transformation into step-counting version, identify composition
of known complexities, rule-based

Rosendahl (1986) – first-order LISP, call-by-value
derives time-bound function and generalizes to sets of inputs

Hickey & Cohen (1988): Performance Compiler – C-like

Flajolet et al. (1990): $\Lambda\Upsilon\Omega$ – average-case complexity

Calculi For Call-by-Need

General question: Do we need to evaluate the argument?

Wadler (1989) – first-order functions
Projections

Bjerner (1989) – first-order functions
Demand analysis (i.e. strictness analysis)

Sands (1990, 1995) – higher-order functions
Cost closures, cost equivalence (*à la* bisimulation)

Please Note

Results not provably correct – but could be made so

Not competing with theoriticians – intend for "bulk" complexity analysis in program synthesis

Time Complexity of Functional Programs

Given profiling semantics $\mathcal{S}$, define time complexity of term t to be the number of reduction steps required to reduce t to canonical form.

Notation:

$$t \downarrow w \text{ (in } n)$$

t reduces to canonical form w in at most n steps

Examples:

$1 + 1 + 1 \downarrow 3$ (in 2)

$\lambda x.x * x \downarrow \lambda x.x * x$ (in 0)

Nuprl Profiling Semantics

(canon) $w \downarrow w$ (in 0) $\quad$ (w canonical)

(ap) $$\frac{f \downarrow \lambda x.b \text{ (in } n_1), \quad b[u/x] \downarrow w \text{ (in } n_2)}{\mathtt{ap}(f; u) \downarrow w \text{ (in } n_1 + n_2 + 1)}$$

(comp) $$\frac{p \downarrow k_1 \text{ (in } n_1),\ q \downarrow k_2 \text{ (in } n_2),\ k_1 < k_2,\ u \downarrow w \text{ (in } n_3)}{\mathtt{less}(p; q; u; v) \downarrow w \text{ (in } n_1 + n_2 + n_3 + 1)}$$

(indstep) $$\frac{p \downarrow k > 0 \text{ (in } n_1), \quad u[k/i, \mathtt{ind}(k-1; b; i,z.f)/z] \downarrow w \text{ (in } n_2)}{\mathtt{ind}(p; b; i,z.f) \downarrow w \text{ (in } n_1 + n_2 + 1)}$$

(abs) $$\frac{\mathtt{op}(\bar{u}) == \phi(\bar{u}), \quad \phi(\bar{u}) \downarrow w \text{ (in } n)}{\mathtt{op}(\bar{u}) \downarrow w \text{ (in } n + 1)}$$

Higher-Order Functions

Meta variables

$(\lambda x.x * x)\, m \downarrow \ldots$ (in $2 + 2\,\mathtt{time}(m)$)

A Higher-Order Example

$$twice == \lambda f.\lambda x.f(fx) : (\alpha \to \alpha) \to \alpha \to \alpha$$

Operator $\tau \to (\alpha \to \alpha)$

$$twice\ g \downarrow \lambda x.g(gx) \text{ (in 1)}$$

Function $\tau_1 \to \tau_2 \to \alpha$

$$twice\ g\ t \downarrow g(gt)? \text{ (in } 2 + \mathtt{time}(g(gt))?)$$

$\mathtt{time}(gt) = \phi(\mathtt{time}(g), \mathtt{term}(t))$?

No visible dependency.

Decompose terms into first-order parts

Abstract Terms

$$\text{(cpx)} \quad \frac{t \downarrow w \ (\text{in } n_1)}{\texttt{cpx}(n; t) \downarrow w \ (\text{in } n + n_1)}$$

Abstract Functions

$$\lambda x.\texttt{cpx}(n_x; t_x)$$

$\Rightarrow$ Simplification: integer variable assumption

All free variables are of type integer.

Lists

Lazy Lists

$$map == \lambda L.\lambda f.\texttt{listind}(L; \texttt{[]}; h, t, z.(f\ h) :: z)$$

$$map\ (\lambda x.x + 1)\ 1 :: 2 :: \texttt{[]} \downarrow \\ ((\lambda x.x + 1)\ 1) :: (map\ (\lambda x.x + 1)\ 2 :: \texttt{[]})(\text{in } 3)$$

Cost of constructing list is deferred to destruction

Abstract Lists

$$\texttt{lcpx}(n; e; k) \quad \widehat{=} \quad e :: \texttt{cpx}(n; \texttt{lcpx}(n; e; k-1))$$

Spinge complexity n, canonical element e, length k

$$\text{(lcpx)} \quad \frac{\begin{array}{c} l \downarrow \texttt{lcpx}(n; e; k)\ (\text{in } n_1), \\ f[e/h,\ \texttt{cpx}(n; \texttt{lcpx}(n; e; k-1))/t, \\ \texttt{listind}(\texttt{lcpx}(n; e; k-1); b; h,t,z.f)/z] \downarrow w\ (\text{in } n_2) \end{array}}{\texttt{listind}(l; b; h,t,z.f) \downarrow w\ (\text{in } n_1 + n_2 + 1)}$$

Examples

$$sumprod == \lambda L.\texttt{listind}(L;\ 0;\ h_1,t_1,z_1.z_1 + \texttt{listind}(t_1;\ 1;\ h_2,t_2,z_2.h_2 * z_2))$$

$$sumprod\ \texttt{lcpx}(a; e; m) \downarrow \ldots \quad (\text{in } 2 + 2m + m^2 + (3+m)am/2 + (m-1)m\,\texttt{time}(e)/2)$$

$$mystery == \lambda L.\texttt{listind}(\texttt{listind}(L;\ []; h1,t1,z1.\texttt{listind}(z1;\ []; h2,t2,z2.h1 :: z2));\ 0; h3,t3,z3.h3 + z3)$$

Complexity of $mystery$ $\texttt{lcpx}(a; e; m)$?

Alternative Semantics

Call-by-value

$$\text{(apv)}\quad \frac{f \downarrow \lambda x.b \text{ (in } n_1\text{)}, \quad u \downarrow v \text{ (in } n_2\text{)}, \quad b[v/x] \downarrow w \text{ (in } n_3\text{)}}{\mathtt{ap}(f; u) \downarrow w \text{ (in } n_1 + n_2 + n_3 + 1\text{)}}$$

$$\text{(spreadv)}\quad \frac{p \downarrow \langle p_1, p_2\rangle \text{ (in } n_1\text{)},\ p_1 \downarrow v_1 \text{ (in } n_2\text{)},\ p_2 \downarrow v_2 \text{ (in } n_3\text{)},\ t[v_1/a, v_2/b] \downarrow w \text{ (in } n_4\text{)}}{\mathtt{spread}(p; a,b.t) \downarrow w \text{ (in } n_1 + n_2 + n_3 + n_4 + 1\text{)}}$$

$$\text{(indstepv)}\quad \frac{p \downarrow k > 0 \text{ (in } n_1\text{)},\ \mathtt{ind}(k-1; b; i,z.f) \downarrow v \text{ (in } n_2\text{)},\ f[k/i,\ v/z] \downarrow w \text{ (in } n_3\text{)}}{\mathtt{ind}(p; b; i,z.f) \downarrow w \text{ (in } n_1 + n_2 + n_3 + 1\text{)}}$$

Call-by-need

Convenient profiling semantics, demand analysis, ...

Symbolic Evaluation

Express complexity of program

$$P\colon \tau_1 \to \tau_2 \to \cdots \to \tau_n$$

in terms of free variables of $P\, a_1\, a_2\, \ldots\, a_n$

Symbolic evaluation

conservative generalization of profiling semantics to open terms

$$(\lambda x.x * x)m \downarrow^s \ldots \text{ (in } 2\,\mathtt{time}(m) + 2)$$

Soundness

$$t_{\bar{v}} \downarrow^s w_{\bar{v}} \text{ (in } n_{\bar{v}}) \quad \text{iff} \quad t_{\bar{u}} \downarrow w_{\bar{u}} \text{ (in } n_{\bar{u}})$$

for all type-correct instantiations $\bar{u}$ of $\bar{v}$.

Symbolic Arithmetic

(var) $x \downarrow x^*$ (in $\mathtt{time}(x)$) (x of type $\mathbb{Z}$)

$$\text{(arith)} \quad \frac{k_1 \downarrow k_1' \text{ (in } n_1), \quad k_2 \downarrow k_2' \text{ (in } n_2)}{\mathtt{add}(k_1; k_2) \downarrow \mathtt{add}^*(k_1'; k_2') \text{ (in } n_1 + n_2 + 1)}$$

Symbolic Cases

$$\texttt{less}(k_1; k_2; t_l; t_r) \begin{cases} t_l \to t_l' \to t_l'' \to \ldots \\ t_r \to t_r' \to t_r'' \to \ldots \end{cases}$$

Need to analyze both computation paths $\Rightarrow$ retain both values

Indeterminate terms

Instantiations of $\texttt{indet}(t_l; t_r)$ are instantiations of t_l and t_r.

$$\text{(comp)} \quad \frac{k_1 \downarrow k_1' \text{ (in } n_1), \quad k_2 \downarrow k_2' \text{ (in } n_2), \quad \texttt{indet}(u; v) \downarrow w \text{ (in } n_3)}{\texttt{less}(k_1; k_2; u; v) \downarrow w \text{ (in } n_1 + n_2 + n_3 + 1)}$$

$\texttt{indet}$ is neither canonical nor non-canonical, but transparent:

$$\text{(indet)} \quad \frac{\begin{array}{c} p \downarrow \texttt{indet}(p_l; p_r) \text{ (in } n_1), \\ \texttt{op}(p_l; \bar{u}) \downarrow w_l \text{ (in } n_l), \ \texttt{op}(p_r; \bar{u}) \downarrow w_r \text{ (in } n_r) \end{array}}{\texttt{op}(p; \bar{u}) \downarrow \texttt{indet}(w_l; w_r) \text{ (in } n_1 + \texttt{tmax}(n_l; n_r))} \quad \text{(meta rule)}$$

No elimination rule for $\texttt{indet}$.

Symbolic Recursion

Problem:
Cannot unfold `ind` definiton if principal argument is symbolic

Computational Skeleton

$$t : \mathbb{Z} \times \mathbb{Z} \quad \longrightarrow \quad \mathtt{cpx}(\rho; \langle \mathtt{cpx}(\rho_1; \mathtt{Ax}); \mathtt{cpx}(\rho_2; \mathtt{Ax}) \rangle)$$

$$u : \mathbb{Z} \to \mathbb{Z} \quad \longrightarrow \quad \mathtt{cpx}(\rho; \lambda x.\mathtt{cpx}(\rho_{\mathtt{a}}(x); \mathtt{Ax}))$$

Representatives

$$\mathsf{rep}_m(t : \alpha) == \mathtt{cpx}(\rho(m); R(\alpha, \epsilon))$$

$$R(\alpha, d) := \begin{cases} \lambda x.\mathtt{cpx}(\rho_{d\cdot\mathtt{a}}(\bar{u}; x); R(\beta, d\cdot\mathtt{a})) & \text{if } \alpha = \mathbb{Z} \to \beta \\ \langle \mathtt{cpx}(\rho_{d\cdot 1}(\bar{u}); R(\beta, d\cdot 1)), & \\ \qquad \mathtt{cpx}(\rho_{d\cdot 2}(\bar{u}); R(\gamma, d\cdot 2)) \rangle & \text{if } \alpha = \beta \times \gamma \\ \mathtt{lcpx}(\rho_{d\cdot\mathtt{t}}(\bar{u}); R(\beta, d\cdot\mathtt{e}); \rho_{d\cdot\mathtt{s}}(\bar{u})) & \text{if } \alpha = \beta\ \mathtt{list} \\ \mathtt{Ax} & \text{if } \alpha = \mathbb{Z} \end{cases} .$$

Representative = decomposition of term into first-order fragments

If all ρ are known, term t and its representative $\mathrm{rep}_m(t)$ are indistinguishable w.r.t. running time.

Exception: whenever the value of an `ind` term matters, e.g. an `ind` term as the principal argument of another `ind` term.

Symbolic induction

$$t_k == \mathtt{ind}(k; b; i, z.f) \downarrow \ldots \text{ (in } \rho(k))$$

Main idea:

Unfold recursion, but evaluate $\mathrm{rep}_{k-1}(t)$ in lieu of t_{k-1} for z

$\Rightarrow$ functional dependency between $\rho(k)$ and $\rho(k-1)$

$$\begin{aligned}
\rho(k) &= \mathtt{time}(t_k) \\
&= 1 + \mathtt{time}(f[k/i, t_{k-1}/z]) \\
&= 1 + \mathtt{time}(f[k/i, \mathtt{cpx}(\rho(k-1), t')/z]) \\
&= 1 + \varphi(\rho(k-1))
\end{aligned}$$

Yields recurrence relation in k:

$$\rho(k) = \begin{cases} \mathtt{time}(f[k/i, \mathtt{cpx}(\rho(k-1), t')/z]) & \text{for } k > 0 \\ \mathtt{time}(b) & \text{for } k = 0 \end{cases}$$

But what is t'?

The type of t will tell us.

1. $t : \mathbb{Z}$
Done: $t' = \texttt{Ax}$

2. $t : \beta \times \gamma \;\Rightarrow t' = \langle t_1; t_2 \rangle$
By construction, we know $t_i = \texttt{cpx}(\rho_i; \ldots)$
Repeat analysis for t_1, t_2 to derive ρ_1, ρ_2

3. $t : \beta \to \gamma \;\Rightarrow t' = \lambda x.b$
By construction, $b = \texttt{cpx}(\rho_{\texttt{a}}(x); \ldots)$ with x free
Repeat analysis for b to derive $\rho_{\texttt{a}}$
Note: $\rho_{\texttt{a}}$ has additional parameter x!

4. $t : \beta$ `list`
Analyze loop body to derive spine cpx, canonical element, and length $\Rightarrow t' = \texttt{lcpx}(\ldots)$ and done

Soundness

To show:

$$t_{\bar{v}} \downarrow^{s} w_{\bar{v}} \text{ (in } n_{\bar{v}}) \;\Rightarrow\; \forall \bar{u}.\mathsf{type}(\bar{u}) = \mathsf{type}(\bar{v}) \supset t_{\bar{u}} \downarrow w_{\bar{u}} \text{ (in } n_{\bar{u}})$$

Proof by induction on derivation of $\downarrow^{s}$:

(var)

Trivial by definition of x^{*} and $\mathtt{time}(x)$.

(comp) $$\frac{k_1 \downarrow^s k_1' \text{ (in } n_1), \quad k_2 \downarrow^s k_2' \text{ (in } n_2), \quad \texttt{indet}(u; v) \downarrow^s w \text{ (in } n_3)}{\texttt{less}(k_1; k_2; u; v) \downarrow^s w \text{ (in } n_1 + n_2 + n_3 + 1)}$$

By IH, $\widehat{k}_i \downarrow \widehat{k}_i'$ (in $\widehat{n}_i$) for any instantiation $\widehat{k}_i, \widehat{n}_i$ of k_i, n_i.

Assume $\widehat{k}_1 < \widehat{k}_2$ and $\widehat{u} \downarrow \widehat{w}$ (in $\widehat{n}$) for instantiations $\widehat{u}, \widehat{w}$ and some $\widehat{n}$. Then $\texttt{less}(\widehat{k}_1; \widehat{k}_2; \widehat{u}; \widehat{v}) \downarrow \widehat{w}$ (in $\widehat{n}_1 + \widehat{n}_2 + \widehat{n} + 1$).

$\widehat{u}$ is instantiation of $\texttt{indet}(u; v)$, so by IH $\widehat{u} \downarrow \widehat{w}$ (in $\widehat{n}_3$) for any instantiation $\widehat{n}_3$.

If $\widehat{n} \geq \widehat{n}_3$, we're done. Other case left as exercise.

q.e.d.

Simplification

Process ...

1. raw complexity expression from symbolic evaluator
2. system of recurrence equations

Subroutines

1. Some built-in simplification rules and recurrence solvers
2. Approximation heuristics for `tmax` terms
3. Link to Mathematica CAS for term simplification and recurrence solving

Approximations

`tmax` terms are ubiquitous, cannot be eliminated

Typical nasty instance:

$$r(m) = \texttt{tmax}(2r(m-1); m)$$

If recurrence solving becomes stuck, apply approximation with increasing force:

0 none

1 $\texttt{tmax}(\sum_{j=1}^{n} a_j u_j + a_0; \sum_{j=1}^{n} b_j u_j + b_0)$
$\mapsto \sum_{j}^{n} \texttt{tmax}(a_j; b_j) u_j + \texttt{tmax}(a_0, b_0)$
where $a_j, b_j > 0$ for $j = 1, \ldots, n$

2 $\texttt{tmax}(\sum_{j=1}^{n} a_j u_j + a_0; \sum_{j=1}^{n} b_j u_j + b_0)$
$\mapsto \sum_{j}^{n} \texttt{tmax}(a_j; b_j) u_j + \texttt{tmax}(a_0, b_0)$

3 $\texttt{tmax}(u; v) \mapsto u + v$

Mathematica Link

Linear equations with constant coefficients

$$r(m,x,\ldots,z) = \begin{cases} \sum_i a_i\, r(m-1, x+b_{xi}, \ldots, z+b_{zi}) & \\ \qquad +q(m,x,\ldots,z) & \text{for } m > 0 \\ p(x,\ldots,z) & \text{for } m = 0 \end{cases}$$

All CAS use characteristic function approach $\sum r(j)\, x^j$ for recurrence solving $\Rightarrow$ not suitable for multi-variable recurrences

Operator approach (Levy & Lessmann, 1961)

$$\begin{aligned} \Xi_\nu \varphi &:= \varphi[\nu + 1/\nu] \\ \Delta_\nu \varphi &:= \varphi[\nu + 1/\nu] - \varphi \end{aligned}$$

Δ_x is discrete analogue of $\partial/\partial x$

Ξ, Δ are distributive, commutative, and satisfy

$$\begin{aligned} \Xi_\nu &= \Delta_\nu + 1 \\ \Xi^{p+q} &= \Xi^p \Xi^q \\ \Delta^{p+q} &= \Delta^p \Delta^q \end{aligned}$$

Usual algebraic identities hold:

$$(\Delta + 1)^2 = \Delta^2 + 2\Delta + 1$$

Can define formal power series:

$$e^{\Xi} = 1 + \Xi + \Xi^2/2! + \Xi^3/3! + \cdots$$

Solving the Recurrence

Step 1: Rewrite as operator equation

$$r(m, x, \ldots, z) = \sum_i a_i \; r(m-1, x+b_{xi}, \ldots, z+b_{zi}) + q(m, x, \ldots, z)$$

$$\Downarrow$$

$$\Xi_m r(m, x, \ldots, z) = \sum_i \left(a_i \; \prod_\nu \Xi_\nu^{b_{\nu i}} \right) r(m, x, \ldots, z) + q(m+1, x, \ldots, z)$$

Also shift from $m/m-1$ to $m+1/m$.

Actually: Redefine Ξ_ν to be $[\nu + b_{\nu i}/\nu]$.

Step 2: Find particular solution

Substitute $\Delta + 1$ for Ξ and move ops to LHS of equation:

$$\Phi\, r_p(m, x, \ldots, z) = q(m+1, x, \ldots, z)$$

for some polynomial $\Phi = \Phi(\Delta_m, \Delta_x, \ldots, \Delta_z)$.

Formally solve for r_p:

$$r_p(m, x, \ldots, z) = \Phi^{-1} q(m+1, x, \ldots, z)$$

Pull out constant factor α of Φ:

$$r_p(m, x, \ldots, z) = 1/\alpha\ (1 - (\underbrace{1 - \Phi/\alpha}_{\tilde{\Psi}}))^{-1} q(m+1, x, \ldots, z)$$

Formal power series for $1/(1-\Psi)$:

$$\begin{aligned} r_p(m,x,\ldots,z) &= 1/\alpha\ (1+\Psi+\Psi^2+\cdots)\ q(m+1,x,\ldots,z) \\ &= 1/\alpha\ (1+\Psi+\Psi^2+\cdots+\Psi^k)\ q(m+1,x,\ldots,z) \end{aligned}$$

for some finite k since $\Psi^j\, q = 0$ for sufficiently large j.

Example:

$$r(m,x,y) = \begin{cases} a\, r(m-1,x+1,y) + r(m-1,x,y+2) & \\ \qquad +mx+y & \text{for } m>0 \\ xy & \text{for } m=0 \end{cases}$$

Step 1 (redefine $\Xi_y\varphi$ to be $\varphi[y+2/y]$):

$$\Xi_m r = (a\Xi_x + \Xi_y)r + mx + x + y$$

Step 2:

$$(\Delta_m - a\Delta_x - \Delta_y - a)r = mx + x + y,$$

Formal solution:

$$r = -1/a \left(1 - (\Delta_m/a - \Delta_x - \Delta_y/a)\right)^{-1} mx + x + y$$

Expand $(1-\Psi)^{-1}$ up to Ψ^2 and simplify

$$r_p(m,x,y) = \frac{1}{-a^2}\left(x + a((m+1)(x-1)+y) - 4\right)$$

$\Rightarrow$ particular solution for $m > 0$

Step 3: Find homogeneous solution

Homogeneous solution r_h satisfies

$$\Xi_m r(m, x, \ldots, z) = \sum_i \left(a_i \prod_\nu \Xi_\nu^{b_{\nu i}} \right) r(m, x, \ldots, z) \text{ for } m > 0$$

Then general solution $r = r_p + \alpha\, r_h$ for $\alpha \in \mathbb{R}$
Case $m = 0$ becomes boundary condition

$$\begin{aligned} r_h(m, x, \ldots, z) &= \Psi\, r_h(m-1, x, \ldots, z) \\ &= \Psi^2\, r_h(m-2, x, \ldots, z) \\ &= \ldots \\ &= \Psi^m\, r_h(0, x, \ldots, z) \\ &= \Psi^m\, A(x, \ldots, z) \end{aligned}$$

A arbitrary, pick $A = p(x, \ldots, z) - r_p(0, x, \ldots, z) \Rightarrow$ satisfies boundary condition for $\alpha = 1$

To determine:

$$r_h(m, x, \ldots, z) = \Psi^m \, A_0(x, \ldots, z)$$

$$\Psi = \psi_1 + \psi_2 + \cdots + \psi_k, \; \psi_i = \Xi_{\nu_1} \Xi_{\nu_2} \ldots \Xi_{\nu_m}$$

Use binomial expansion to eliminate operators:

$$\begin{aligned} r_h(m, x, \ldots, z) &= (\psi_1 + \Psi_1)^m A_0(x, \ldots, z) \\ &= \sum_{i=0}^{m} \binom{m}{i} \psi_1^i (\Psi_1^{m-i} A_0(x, \ldots, z)) \\ &= \cdots \\ &= \sum_{i=0}^{m} \binom{m}{i} \psi_1^i A_{m-1}(x, \ldots, z) \\ &= \sum_{i=0}^{m} \binom{m}{i} A_m(x, \ldots, z) \end{aligned}$$

Note: $\Xi_x^j \Xi_y^k \, f(x, y) = f(x + j, y + k)$

Rely on Mathematica to eliminate sum from last line

Example (conclusion)

$$\begin{aligned}
r_h(m,x,y) &= (a\Xi_x + \Xi_y)^m (xy - r_p(m,x,y)) \\
&= \sum_{i=0}^{m} \binom{m}{i} (a\Xi_x)^i (\Xi_y)^{m-i} (xy - r_p(m,x,y)) \\
&= \sum_{i=0}^{m} \binom{m}{i} (a\Xi_x)^i (x(y+2m-2i) - r_p(m,x,y+2m-2i)) \\
&= \sum_{i=0}^{m} \binom{m}{i} a^i \Big((x+i)(y+2m-2i) - r_p(m,x+i,y+2m-2i) \Big)
\end{aligned}$$

With Mathematica:

$$r(m,x,y) = r_p(m,x,y) + r_h(m,x,y)$$

Examples

Example 1: Maximum Segment Sum

Given list of integers $a_1 :: \cdots :: a_n$, find maximum segment sum:

$$M = \max_{1 \le i,j \le n} \sum_{k=i}^{j} a_k$$

Straight-forward imperative solution: $O(n^2)$
Clever imperative solution by Gries: $O(n)$

```
l, m := 0
for i := 1 to n do
    l := max(l + a[i], 0)
    m := max(m, l)
```

Result of ACAp System:

```
M> let mss = f_of_thm ‘maxsegsum‘ ;;
M> fcc mss ⌜lcpx(0;0;m)⌝ ;;
5 + 17 * m * 1/2 + 3 * m * m * 1/2 : term
```

Actual running time for worst-case list:

m:	0	1	2	3	4	5	6	7	8	9	...
$\rho(m)$:	4	14	27	43	62	84	109	137	168	202	...

$\Rightarrow \rho(m) = 4 + 17m/2 + 3m^2/2$

Manual analysis:

$$T(n) = 2\,T(n-1) + c$$

So why isn't it exponential?

Example 2: Minimum via Sorting

Insertion sort

$$\texttt{ielem}(x; L) \;==\; \texttt{listind}(L;\; x :: \texttt{[]}; h,t,z.\; \texttt{ifless}(x;\; h;\; x :: h :: t;\; h :: z))$$
$$\texttt{isort}(L) \;==\; \texttt{listind}(L;\; \texttt{[]};\; h,t,z.\texttt{ielem}(h; z)),$$

Sort list of length m:
Call-by-value: $O(m^2)$
Call-by-name: $[O(m^2)]$

Minimum as $\texttt{hd}(\texttt{isort}(L))$:
Call-by-value: $O(m^2)$
Call-by-name: $O(m)$

Conclusion

Abstract cost model for Nuprl call-by-name semantics

Central to calculus: abstract terms `cpx`, `lcpx`

Symbolic evaluation of open terms

ACA*p* system derives multi-variable recurrence equations from program

Solves recurrences using Mathematica

Currently supports first-order functions, lazy lists, certain higher-order functions

Future Work

Extend to more classes of recurrence equations
Derive formal proofs of time complexity using reflection
Make more interactive
General recursion

Higher-order functions

Discard integer variable assumption
Decompose free variables into first-order parts, similar to `ind` terms

$$f : \mathbb{Z} \to \mathbb{Z} \quad \longrightarrow \quad \mathtt{cpx}(f_{\mathtt{o}}, \lambda x.\mathtt{cpx}(f_{\mathtt{a}}(x); \mathtt{Ax}))$$

Higher-Order Functions

Example:

$$\texttt{ind}(k; \lambda f.f0; i,z.\lambda f.z(\lambda x.x+1)) \colon (\mathbb{Z} \to \mathbb{Z}) \to \mathbb{Z} \to \mathbb{Z}$$

Generates "higher-order" recurrence equation:

$$\begin{aligned}\rho_{\mathtt{a}}(m, f, f_{\mathtt{a}}) \;=\; & \rho_{\mathtt{a}}(m-1, f(m-1), 1 + f_{\mathtt{a}}(m-1, x)) \\ & + \rho_{\mathtt{a}}(m-1, 0, 1 + f(m-1) + f_{\mathtt{a}}(m-1, x+1)) + c\end{aligned}$$

Current operator approach fails since

$$\Xi_\nu\, \varphi := \varphi[f(\nu)/\nu]$$

is no longer commutative!