Predicative Copying and Polynomial Time

Samson Abramsky

Background: Implicit Computational Complexity

More specifically, logics and type theories which delineate complexity classes.

E.g.

Light Linear Logic	Girard
SLR, Space-bounded calculi	Hofmann, Schwichtenberg etc. (after Bellantoni–Cook, Leivant)

Some salient features of our approach:

- Type-free, purely applicative systems (prefigured by LLL). Complexity bounds **not** enforced by restricting the primitive recursion scheme — there are no types!
- Simple, both in syntactic formulation, and the underlying concepts.
- Simple notion of model.
- Simple extension to non-determinism.

Combinatory Logic

Application: $x \cdot y$.
(We write $xy_1 \cdots y_n$ for $(\cdots(x \cdot y_1) \cdots) \cdot y_n)$).

Combinators **S**, **K**:

$$\begin{aligned} \mathbf{S}xyz &= xz(yz) \\ \mathbf{K}xy &= x \end{aligned}$$

(We can define $\mathbf{I} \equiv \mathbf{SKK}$, satisfying $\mathbf{I}x = x$).

Functional or 'bracket' abstraction:
$[x]t$ such that $([x]t)u = t[u/x]$.

$$\begin{aligned} [x]x &= \mathbf{I} \\ [x]y &= \mathbf{K}y \qquad (x \neq y) \\ [x]tu &= \mathbf{S}([x]t)([x]u) \end{aligned}$$

The Curry Combinators: B, C, K, W

$$\begin{aligned} \mathbf{B}xyz &= x(yz) \\ \mathbf{C}xyz &= xzy \\ \mathbf{W}xy &= xyy \end{aligned}$$

Principal types:

I	:	$\alpha \to \alpha$	Axiom
B	:	$(\beta \to \gamma) \to (\alpha \to \beta) \to \alpha \to \gamma$	Cut
C	:	$(\alpha \to \beta \to \gamma) \to \beta \to \alpha \to \gamma$	Exchange
K	:	$\alpha \to \beta \to \alpha$	Weakening
W	:	$(\alpha \to \alpha \to \beta) \to \alpha \to \beta$	Contraction

Curry's analysis of substitution is close to Gentzen's analysis of proofs.

Sub-structural Logic

The **BCK** combinators support **affine bracket abstraction**: $[x]t$ where x **occurs at most once** in t.

We can define $\mathbf{I} \equiv \mathbf{CKK}$.

$$\begin{array}{lcll} [x]x & = & \mathbf{I} & \\ [x]t & = & \mathbf{K}t, & (x \notin FV(t)) \\ [x]tu & = & \mathbf{C}([x]t)u, & (x \in FV(t) \setminus FV(u)) \\ [x]tu & = & \mathbf{B}\,t\,([x]u), & (x \in FV(u) \setminus FV(t)) \end{array}$$

Higher-order defining equations (after D. Turner)

We can use bracket abstraction to define functions by higher-order curried equations.

E.g. we can define F by

$$Fxyz = t$$

meaning $F \equiv [x][y][z]t$, satisfying

$$Fabc = t[a/x, b/y, c/z].$$

Example: defining S from BCKW.

Linear version of **S**:

$$\mathbf{S}'fgxy = fx(gy)$$

Then we recover **S** by

$$\mathbf{S}fgx = \mathbf{W}(\mathbf{S}'fg)x$$

Computational Power

In Combinatory Logic, all partial recursive functions are **numeralwise representable**.

This means that we can define numeral systems representing each number n as a term $\bar{n}$ such that, for every recursive function

$$f : \mathbb{N} \longrightarrow \mathbb{N}$$

there is a term t satisfying:

$$\forall n \in \mathbb{N}.\, t\bar{n} = \bar{m} \iff f(n) = m.$$

The GAP

By (extreme) contrast, in **BCK** logic:

$$\begin{aligned} \mathbf{B}xyz &\rightarrow x(yz) \\ \mathbf{C}xyz &\rightarrow xzy \\ \mathbf{K}xy &\rightarrow x \end{aligned}$$

every reduction step strictly decreases the size of the term, so all terms normalize in linear time!

This highlights **the computational power of copying**:

BCKW	universal computational power
BCK	linear time reduction

This is a big gap: how can we find extra structure to expose the intermediate possibilities?

Affine Combinatory Logic

(Combinatory logic view of $\multimap$, ! fragment of Affine Logic (which is Linear Logic + Weakening))

The key idea: extend **BCK** logic with an additional unary operation which we write as !.

Logically, application is Modus Ponens:

$$\frac{t : \alpha \to \beta \qquad u : \alpha}{tu : \beta}$$

Affine application, characterized by **BCK**, is the logic of affine implication $\multimap$.

! is Necessitation:

$$\frac{t : \alpha}{!t : !\alpha} \qquad \frac{t : \alpha}{\Box t : \Box \alpha}$$

The following combinators give ! the structure of an S4 modality:

$$\begin{aligned} \mathbf{F}\,!x\,!y &= !(xy) \\ \mathbf{D}\,!x &= x \\ \delta\,!x &= !!x \end{aligned}$$

Principal types:

$\mathbf{F} : !(\alpha \multimap \beta) \multimap !\alpha \multimap \beta$	$\Box(A \to B) \to \Box A \to \Box B$
$\mathbf{D} : !\alpha \multimap \alpha$	$\Box A \to A$
$\delta : !\alpha \multimap !!\alpha$	$\Box A \to \Box\Box A$

The additional Linear Logic idea is that its modality signifies **copyability**. We have a combinator $\mathbf{W}^!$:

$$\mathbf{W}^!\,x\,!y = x\,!y\,!y$$

$$\mathbf{W}^! : (!\alpha \multimap !\alpha \multimap \beta) \multimap !\alpha \multimap \beta$$

Interpretation of Standard Combinatory Logic

We can interpret standard CL into Affine CL. We interpret standard application by

$$x \cdot_s y \equiv x \cdot_a !y$$

We can then define the standard combinators with respect to this application from the combinators of Affine CL.

So again in Affine CL all partial recursive functions are representable.

But we now have some extra structure to play with ...

A Variant

Replace the $\mathbf{W}^!$ combinator by a family

$$\{\mathbf{W}^n \mid n > 0\}$$

with defining equations

$$\mathbf{W}^n\, x\, !y = x\, \underbrace{y \cdots y}_{n}$$

This variant is equivalent to Affine CL as previously presented: in particular, we can define

$$\mathbf{W}^!\, x\, z = \mathbf{W}^2\, x\, (\delta z)$$

since then

$$\mathbf{W}^!\, x\, !y = \mathbf{W}^2\, x\, (\delta\, !y) = \mathbf{W}^2\, x\, !!y = x\, !y\, !y.$$

However, what if we remove δ?

Then ! becomes a 'copyability resource' which gets **consumed** when we apply a $\mathbf{W}^n$. We can think of $!t$ as a promissory note:

I promise to give the bearer on demand n copies of t, for any n

An occurrence of $\mathbf{W}^n$ is used to cash this promissory note in.

By contrast, note the **recursivity** or **unboundedness** or **impredicativity** of $\mathbf{W}^!$:

$$\mathbf{W}^! \, x \, !y = x \, !y \, !y$$

We make two copies of $!y$, which themselves are copyable ...

So we are replacing unbounded or impredicative copying by a predicative version.

Predicative Combinatory Logic (PCL)

Combinators $\mathbf{B}$, $\mathbf{C}$, $\mathbf{K}$, $\mathbf{F}$, $\mathbf{W}^n$ $(n > 0)$.

(N.B. We don't need $\mathbf{D}$, since we can define it using $\mathbf{W}^1$).

We give the reduction rules for these combinators:

$$\begin{array}{lcl} \mathbf{B}xyz & \to & x(yz) \\ \mathbf{C}xyz & \to & xzy \\ \mathbf{K}xy & \to & x \\ \mathbf{F}\,!x\,!y & \to & !(xy) \\ \mathbf{W}^n\,x\,!y & \to & x\,\underbrace{y\cdots y}_{n} \end{array}$$

PCL is an orthogonal term-rewriting system, hence confluent (*i.e.* the Church-Rosser property holds).

Analysis of PCL

We introduce some measures on terms:

Size (number of leaves in the term tree):

$$s(c) = 1 \qquad s(tu) = s(t) + s(u) \qquad s(!t) = s(t)$$

Depth (maximum nesting depth of !'s):

$$d(c) = 0 \qquad d(tu) = \max(d(t), d(u)) \qquad d(!t) = 1 + d(t)$$

Width:

$$w(t) = \max(\{n \mid \mathbf{W}^n \text{ occurs in } t\} \cup \{1\})$$

Weight: Firstly, define $P_t(x)$, a polynomial in the variable x:

$$P_c(x) = 1 \qquad P_{tu}(x) = P_t(x) + P_u(x) \qquad P_{!t}(x) = x \times P_t(x)$$

Then define the weight: $\|t\| = P_t(w(t))$.

Proposition 1 *For all t: $\|t\| \leq w^d s$, where $w = w(t)$, $s = s(t)$, $d = d(t)$.*

Thus the weight is **linear in the size**, **polynomial in the width**, and **exponential in the depth**.

Proposition 2 *If $t \to u$, then $\|t\| > \|u\|$.*

Thus every term is Strongly Normalizing, with the length of all reduction sequences bounded by $\|t\|$.

Proposition 3 *Reduction to normal form of PCL terms can be simulated by a Turing machine with polynomial overhead.*

[Go via the RAM model.]

Consequences for numeralwise representability

Suppose we have a system of representations (say of binary numerals) as terms, which is of **bounded depth**, *i.e.* for some $d_0 \geq 1$, for all n,

$$d(\bar{n}) \leq d_0,$$

and also $s(\bar{n}), w(\bar{n}) \in O(|n|)$. Then **any** function numeralwise representable in PCL with respect to this system is polynomial time.

Indeed, suppose t n.w.-represents f, then for all n,

$$\|t\bar{n}\| \leq w^d s$$

where $d = \max(d(t), d_0)$, and s, w are $O(|n|)$.

High-level notations for PCL

- Affine higher-order defining equations: we can define

$$Fx_1 \ldots x_n = t$$

  where each x_i occurs at most once in t.

- Conditionals. The booleans are defined as usual, $\mathsf{tt}\, xy = x$, $\mathsf{ff}\, x\, y = y$, and then the conditional

$$\textbf{if } b \textbf{ then } t \textbf{ else } u$$

  where t and u can share variables, but must be disjoint from b, can be defined as

$$b([x_1] \ldots [x_n]t)([x_1] \ldots [x_n]u)x_1 \ldots x_n$$

  where $x_1, \ldots, x_n$ are the variables shared by t and u.

- !-lifting. We define $[x]^i t$, $i \geq 0$, inductively on i.

$$[x]^0 t = [x]t, \qquad [x]^{i+1} t = \mathbf{F}\,!([x]^i t).$$

Then, writing $!^i u$ for $\underbrace{!\cdots!}_{i} u$,

$$[x]^i\, t\, !^i u = !^i(t[u/x]).$$

- Copying abstraction.

$$\phi\, !(x_1, \ldots, x_n) = t$$

$$\phi \equiv \mathbf{W}^n u \textbf{ where } u x_1 \ldots x_n = t.$$

This satisfies

$$\phi\, !v = t[v/x_1, \ldots, v/x_n].$$

Expressiveness of PCL

The key step in showing that PCL can represent all PTIME functions is to show how **polynomial-length iterations** can be represented.

Unary numerals in PCL.

$$\bar{n}\,!(x_1, \ldots, x_n)y = x_1(\cdots(x_n y)\cdots).$$

This is a **very** weak numeral system — we can't even define successor! Nevertheless ...

Given a polynomial $P(X)$ of degree k with coefficients in $\mathbb{N}$, and a term u, we want to define a term $\mathsf{iter}_{P,u}$ such that, for all $n \in \mathbb{N}$,

$$\mathsf{iter}_{P,u}\, \underbrace{\bar{n} \cdots \bar{n}}_{k}\, x = u^{P(n)} x$$

where

$$u^0 x = x, \qquad\qquad u^{i+1} x = u(u^i x).$$

This is a very weak notion of iteration — not uniform in u.
Nevertheless ...

For simplicity, we concentrate on the case $P(X) = X^k$. We define a term t_{X^k} such that, for all $n \in \mathbb{N}$,

$$t_{X^k} \underbrace{\bar{n} \cdots \bar{n}}_{k} !^k f\, x = f^{n^k} x.$$

Given this term, we can define

$$\mathsf{iter}_{X^k,u} u_1 \cdots u_k = t_{X^k} u_1 \cdots u_k\, !^k u.$$

The general shape for a polynomial $P = \Sigma_{i=0}^{k} c_i X^k$ is

$$t_P \underbrace{\bar{n} \cdots \bar{n}}_{k} \underbrace{f \cdots f}_{c_0} \cdots \underbrace{!^k f \cdots !^k f}_{c_k}$$

We define t_{X^k} by induction on $k \geq 1$.

$$t_X uv = uv.$$

$$t_{X^{k+1}} u_1 \cdots u_k u_{k+1} v = u_{k+1}(t_{X^k} u_1 \cdots u_k(\theta_k v)\, !\mathbf{I})$$

where

$$\theta_k = [v]^k(\mathbf{F}(([w]^1[g][y]w(gy))v))$$

Illustration: $k = 2$

$$\begin{aligned} t_{X^2}\bar{n}\bar{n}\,!!f &= \bar{n}(\bar{n}(\theta_1\,!!f)\,!\mathbf{I}) \\ &= \bar{n}(\bar{n}\,!(\mathbf{F}\,!([g][y]f(gy)))\,!\mathbf{I}) \\ &= \bar{n}(\phi^n\,!\mathbf{I}) \end{aligned}$$

where $\phi \equiv \mathbf{F}\,!([g][y]f(gy))$.

$$\phi^n\,!\mathbf{I} = ![y]f^n(\mathbf{I}y) = ![y]f^n y$$

$$\mathbf{F}\cdots!(\mathbf{F}\,!(\ldots)(\mathbf{F}\,!(\ldots)\,!\mathbf{I}))\cdots$$

The ! percolates up the nested sequence of applications.

Finally,

$$\bar{n}\,!([y]f^n y)x = f^{n^2}x.$$

High-level structure of the representation

Suppose we are given a polynomial time function represented by a Turing Machine with a bounding polynomial P.

- We can program the transition function of a space-bounded Turing Machine (which never extends the tape) by purely affine means using conditionals and a representation of lists.
- Given a binary numeral for n, we can convert it into an initial configuration for the Turing Machine. We then use a $P(n)$-length iteration to pad the tape out with blanks, making it large enough for the entire computation.
- We then use another $P(n)$-length iteration with the Turing Machine transition function to perform the computation.
- Finally, we can extract the result from the final tape, again by purely affine means.

Non-Determinism

We can extend PCl with non-deterministic choice:

$$t \sqcap u \quad \rightarrow \quad t$$

$$t \sqcap u \quad \rightarrow \quad u$$

The resulting system characterizes NP in the same sense that PCL characterizes P.

Related Work

- Light Linear Logic (Girard). Differences: we get rid of δ, keep **F** and **D**, modify copying. LLL gets rid of **D** and **F**, keeps copying unchanged, and introduces a new connective with its own version of **F** and a pseudo-dereliction from !.
- Soft Linear Logic (Lafont). The same essential ideas in a very different setting — typed, proof-theoretic, graph-rewriting. Our presentation (imho) is much simpler!

Models

In finding models for these restricted calculi, one looks for **positive reasons** for various constructs to be omitted. That is, one is looking for key structural properties which **characterize** PTIME algorithmic processes.

(Analogous to game semantics for PCF etc.)

- For LLL, this seems very subtle.
- For systems like SLR, not even attempted — realizability models simply build the complexity constraints in as an assumption.
- With PCL there is a very simple basic notion of model which already explains the absence of the omitted principles.

As a first approximation, take

$$!A = \prod_{i \in \omega} A^n \qquad\qquad A^n = \underbrace{A \otimes \cdots \otimes A}_{n}.$$

Then the $\mathbf{W}^n$ are just the projections $\pi_n : !A \longrightarrow A^n$.

$$\mathbf{F} : !(A \multimap B) \otimes !A \longrightarrow !B$$

Given the choice by the context of a number of copies of the output, the corresponding number of copies is chosen by the input, and application is performed component-wise.

Moreover, ! is clearly functorial.

However, no map $!A \longrightarrow !A \otimes !A$

$$\prod_k A^k \longrightarrow \prod_n A^n \otimes \prod_m A^m$$

since we have to respond to the choice by the context of an n (or an m) with a choice of k, and no such choice can work for all subsequent choices the context may make for m (or n).

Similarly, there is no map $!A \longrightarrow !!A$

$$\prod_k A^k \longrightarrow \prod_n (\prod_m A^m)^n$$

since we must choose a k in response to the choice by the environment of an n, and then of an m in some tensorial factor i of $(\prod_m A^m)^n$; and no such choice will work for all subsequent choices for m which the context may subsequently make in other factors.

A more refined notion of model, to reflect the idea that 'all copies are identical' (and that we have the same behaviour independent of how many copies are requested), is to take $!A$ to be the **limit** of the diagram with nodes A^n, and maps

$$A^n \cong A^n \otimes I \longleftarrow A^{n+1}$$

using weakening (*i.e.* the fact that the tensor unit is the terminal object), and

$$\hat{\pi} : A^n \xrightarrow{\cong} A^n$$

for each permuation $\pi \in S(n)$.

Standard Game models (e.g. AJM games) have these limits, and hence provide examples of such models.

Main Aim: a Full Completeness Theorem.