Feasible computation on general sets

Arnold Beckmann

(joint work with Sam Buss and Sy Friedman)

Department of Computer Science
College of Science
Swansea University, Wales
UK

Logic Colloquium 2012
Manchester, 13 July 2012

Motivation

Computation on other structures than finite strings:

1. Over the reals:
   Blum, Shub, Smale, and many others
2. Infinite Time Turing Machine:
   Deolalikar, Hamkins, Schindler, Welch, and others
3. Molecular Biology / DNA computing:
   Aldeman, Lipton
4. Quantum Computing:
   Shore

Question: What is a good notion of feasible computation on arbitrary sets?

Characterisations of Polytime on Finite Strings

Notation: ϵ empty word; $w\,i$ = append bit i to word w;
$|w|$ denotes length of w (number of bits.)

Characterisations of f being polytime computable:

1. There exists Turing machine M which on input w computes $f(w)$ with runtime bounded polynomially in $n = |w|$.
2. Cobham's Bounded Recursion on Notation:

$$f(\epsilon, \vec{x}) = g(\vec{x})$$
$$f(y\,i, \vec{x}) = h_i(y, \vec{x}, f(y, \vec{x})) \qquad (i \in \{0,1\})$$

   provided that $f(y, \vec{x}) \leq j(y, \vec{x})$ for all $y, \vec{x}$.
3. Recursion schemes without explicit bounds:
   Leivant, Bellantoni/Cook and others.

"Polytime" for sets

1. Turing Machine:
   Difficult to write an arbitrary set on a tape of length ω.
2. Recursion schemes:
   Cobham: *bounded recursion on notations*
   Leivant: *tired recursion*
   Bellantoni/Cook: *safe recursion*
3. ...

We will adapt Bellantoni/Cook's approach to set functions.

Outline of talk

Recap: Bellantoni-Cook's Characterisation

Define functions on finite binary strings

$$f(x_1, \ldots, x_k \,/\, a_1, \ldots, a_\ell)$$

$x_1, \ldots, x_k$ are the normal inputs, $a_1, \ldots, a_\ell$ the safe inputs to f.

Bellantoni-Cook's class B: Smallest class containing

i) **(Constant)** ϵ (zero-ary)

ii) **(Projection)** $\pi_j^{n,m}(x_1, \ldots, x_n \,/\, x_{n+1}, \ldots, x_{n+m}) = x_j$, for $1 \leq j \leq n + m$.

iii) **(Successors)** $\mathsf{s}_i(-/\, a) = a\, i$, for $i \in \{0, 1\}$

iv) **(Predecessor)** $\mathsf{p}(-/\, \epsilon) = \epsilon, \quad \mathsf{p}(-/\, a\, i) = a$

v) **(Conditional)** $\mathrm{Cond}(-/\, a, b, c) = \begin{cases} b & \text{if } a = d\, 1 \\ c & \text{otherwise.} \end{cases}$

Recap: Bellantoni-Cook's Characterisation

. . . and closed under

vi) **(Predicative Recursion on Notation)**

$$\begin{aligned} f(\epsilon, \vec{x} \,/\, \vec{a}) &= g(\vec{x} \,/\, \vec{a}) \\ f(z\,i, \vec{x} \,/\, \vec{a}) &= h_i(z, \vec{x} \,/\, \vec{a}, f(z, \vec{x} \,/\, \vec{a})) \quad i \in \{0,1\} \end{aligned}$$

Spirit: The recursion argument has to be normal, while the "previous value" of the recursion is placed into a safe position.

vii) **(Safe Composition)** $f(\vec{x} \,/\, \vec{a}) = h(\vec{r}(\vec{x} \,/ -) \,/\, \vec{t}(\vec{x} \,/\, \vec{a}))$

(Note: no typo, the r_j's don't have any safe arguments!)

Spirit: When composing functions be careful not to allow safe inputs to be copied into normal positions.

Examples

Concatenation of words $\oplus(x\,/\,a) = a \ast\ast x$ is in the class, by one predicative recursion:

$$\oplus(\epsilon\,/\,a) = a$$
$$\oplus(x\,i\,/\,a) = \mathsf{s}_i(-/\ \oplus(x\,/\,a)) = \oplus(x\,/\,a)\,i$$

Observe $|\oplus(x\,/\,a)| = |x| + |a|$.

Then "smash" $\odot(x, y\,/-)$ is in the class, by a second predicative recursion:

$$\odot(\epsilon, y\,/-) = \epsilon$$
$$\odot(x\,i, y\,/-) = \oplus(y\,/\ \odot(x, y\,/-)) = \odot(x, y\,/-) \ast\ast y$$

Observe $|\odot(x\,/\,a)| = |x| \cdot |a|$.

Examples

But "exponentiation" is *not* in the class!

To define "exponentiation" using smash one would need something like

$$E(x\,i, y\,/-) = \odot(y\,/\,E(x, y\,/-)) \qquad (\text{then } |E(x, y\,/-)| = |y|^{|x|})$$

but we only have

$\odot(y, z\,/-)$ (smash of two normal inputs)

and no function

$\odot(y\,/\,z)$

which has z as a safe input.

Another possibility

$$E(x\,i\,/-) = \oplus(E(x\,/-)\,/\,E(x\,/-)) \qquad (\text{then } |E(x\,/-)| = 2^{|x|})$$

again cannot be typed according to existing normal/safe inputs.

Bellantoni-Cook's '92 Results

Lemma (Boundedness)

For any safe recursive function $f(x_1, \ldots, x_k \,/\, a_1, \ldots, a_\ell)$ *there is a polynomial* p *such that*

$$|f(\vec{x} \,/\, \vec{a})| \;\leq\; \max(|\vec{a}|) + p(|\vec{x}|)$$

($|\vec{x}|$ *denotes vector* $|x_1|, \ldots, |x_k|$*; similar* $|\vec{a}|$*.)*

Theorem

Let $f(\vec{x} \,/\, \vec{a})$ *be safe recursive. Then* $f(\vec{x}, \vec{a})$ *is polynomial time computable.*

Theorem

Let be $f(\vec{x})$ *polynomial time computable on finite strings. Then* $f(\vec{x} \,/ -)$ *is Bellantoni-Cook safe recursive.*

Rudimentary Set Functions

The Gandy-Jensen Rudimentary Set Functions are the smallest class containing i) – iii), and being closed under iv) – v):

i) **(Projection)** $\pi_j^n(x_1, \ldots, x_n) = x_j$, for $1 \leq j \leq n$.

ii) **(Difference)** $\mathrm{diff}(a, b) = a \setminus b = \{x \in a \colon x \notin b\}$

iii) **(Pairing)** $\mathrm{pair}(a, b) = \{a, b\}$

iv) **(Union Scheme)**
$f(\vec{x}, y) = \bigcup_{z \in y} g(\vec{x}, z)$

v) **(Composition Scheme)**
$f(\vec{x}) = h(\vec{t}(\vec{x}))$

Examples for Rudimentary Set Functions

- $\mathrm{union}(b) = \bigcup b$ $\quad \left[\mathrm{union}(b) = \bigcup_{z \in b} \pi^1_1(z)\right]$
- $\mathrm{Succ}(a) = a \cup \{a\}$ $\quad \left[\mathrm{Succ}(a) = \mathrm{union}(\mathrm{pair}(a, \mathrm{pair}(a, a)))\right]$
- $\mathrm{Cond}_=(a, b, c, d) = \begin{cases} a & \text{if } c = d \\ b & \text{otherwise.} \end{cases}$

  $\Big[\ \bar{g}(a, c, z) := \bigcup\{a\colon u \in c \setminus z \cup z \setminus c\} = \begin{cases} a & \text{if } z \neq c \\ \emptyset & \text{otherwise} \end{cases}$

  and $g(a, c, z) := a \setminus \bar{g}(a, c, z)$

  Then $\mathrm{Cond}_=(a, b, c, d) = g(a, c, d) \cup \bar{g}(b, c, d).$ $\Big]$
- $\mathrm{Cond}_\in(a, b, c, d) = \begin{cases} a & \text{if } c \in d \\ b & \text{otherwise.} \end{cases}$

  $\Big[\ h(a, c, d) := \bigcup\{g(a, c, z)\colon z \in d\};\ \bar{h}(b, c, d) := b \setminus h(b, c, d),$

  then $\mathrm{Cond}_\in(a, b, c, d) = h(a, c, d) \cup \bar{h}(b, c, d).$ $\Big]$

Primitive Recursive Set Functions

The Primitive Recursive Set Functions are the smallest class containing i) – iii), and being closed under iv) – vi):

vi) **(Primitive Set Recursion Scheme)**
$f(x, \vec{y}) = h(x, \vec{y}, \{f(z, \vec{y}) : z \in x\})$

Examples

Addition, multiplication, exponentiation on ordinals are primitive recursive.

Safe Recursive Set Functions

Idea: Add to Gandy-Jensen rudimentary set functions a safe recursion scheme a la Bellantoni-Cook.

Safe Recursive Set Functions

The Safe Recursive Set Functions are the smallest class containing i) – iii), and being closed under iv) – vi).

i) $\pi_j^{n,m}(x_1, \ldots, x_n \,/\, x_{n+1}, \ldots, x_{n+m}) = x_j$, for $1 \leq j \leq n+m$.

ii) $\mathrm{diff}(-/\, a, b) = a \setminus b$

iii) $\mathrm{pair}(-/\, a, b) = \{a, b\}$

iv) **(Rudimentary Union Scheme)**
$f(\vec{x} \,/\, \vec{a}, b) = \bigcup_{z \in b} g(\vec{x} \,/\, \vec{a}, z)$

v) **(Safe Composition Scheme)**
$f(\vec{x} \,/\, \vec{a}) = h(\vec{r}(\vec{x} \,/ -) \,/\, \vec{t}(\vec{x} \,/\, \vec{a}))$

vi) **(Safe Set Recursion Scheme)**
$f(x, \vec{y} \,/\, \vec{a}) = h(x, \vec{y} \,/\, \vec{a}, \{f(z, \vec{y} \,/\, \vec{a}) \colon z \in x\})$

Examples for Safe Recursive Set Functions

$$\mathsf{Add}(x \,/\, a) = \begin{cases} a & \text{if } x = 0 \\ \mathsf{Succ}(-/ \bigcup \{\mathsf{Add}(z \,/\, a) \colon z \in x\}) & \text{if } x = \mathsf{Succ}(-/ \bigcup x) \\ \bigcup \{\mathsf{Add}(z \,/\, a) \colon z \in x\} & \text{otherwise.} \end{cases}$$

$\alpha + \beta := \mathsf{Add}(\beta \,/\, \alpha)$ defines usual addition on ordinals α, β.

$$\mathsf{Mult}(x, y \,/ -) = \begin{cases} 0 & \text{if } x = 0 \\ \mathsf{Add}(y \,/ \bigcup \{\mathsf{Mult}(z, y \,/ -) \colon z \in x\}) & \text{if } x = \mathsf{Succ}(-/ \bigcup x) \\ \bigcup \{\mathsf{Mult}(z, y \,/ -) \colon z \in x\} & \text{otherwise.} \end{cases}$$

$\alpha \cdot \beta := \mathsf{Mult}(\beta, \alpha \,/ -)$ defines usual multiplication on ordinals α, β.

Bounding Ranks

But ordinal exponentiation is *not* safe recursive:

Theorem

Let f be a safe recursive set function. There is a polynomial q_f such that

$$\mathrm{rank}(f(\vec{x} \,/\, \vec{a})) \quad \leq \quad \max(\mathrm{rank}(\vec{a})) + q_f(\mathrm{rank}(\vec{x}))$$

for all sets $\vec{x}$, $\vec{a}$.

SR Set Functions on Hereditarily Finite Sets

SR functions grow ranks polynomially
$\Rightarrow$ super-exponential bound on sizes of sets for SR set functions.

We can do better:

Example

Ordered pair $(a, b) := \{\{a\}, \{a, b\}\}$.
$\mathrm{Prod}(-/\, a, b) = a \times b = \{(x, y) \colon x \in a, y \in b\}$ is rudimentary.
Let $\mathrm{Sq}(-/\, a) = \mathrm{Prod}(-/\, a, a)$.
Define f by safe recursion as follows:
$f(\emptyset \,/\, a) = a, \quad f(\{d\} \,/\, a) = \mathrm{Sq}(-/\, f(d \,/\, a))$.
Then f is SR, and satisfies

$$\mathrm{card}(f(x \,/\, a)) = \mathrm{card}(a)^{2^{\mathrm{rank}(x)}}$$

SR Set Functions on HF are Dietary

Previous example illustrates "worst case":

Definition

$f(\vec{x} \,/\, \vec{a})$ in *SRSF* is called *dietary* if for some polynomial p,

$$\mathrm{card}(\mathrm{tc}(f(\vec{x} \,/\, \vec{a}))) \quad \leq \quad \mathrm{card}(\mathrm{tc}(\{\vec{x}, \vec{a}\}))^{2^{p(\mathrm{rank}(\vec{x}))}}$$

for all $\vec{x}, \vec{a} \in \mathrm{HF}$.

Theorem

All functions in SRSF are dietary.

Representing Tapes

Let M be a non-deterministic Turing Machine,
and p some polynomial.

Represent tapes as full binary trees using the ordered pair (a, b) with leafs labelled by tape symbols.

Thus: trees of height h represent tapes of length 2^h.

$$f(x/-) \mapsto \{c : c \text{ is a tree of height } p(\mathrm{rank}(x))\}$$

is SR (by repeated squaring.)

Representing Transitions

$$g(x\,/-) \mapsto \{(c,d):\ c,d \in f(x\,/-) \text{ and } d \text{ can be obtained from } c \text{ in } \leq 2^{p(\mathrm{rank}(x))} \text{ many } M\text{-steps } \}$$

Fact: g is SR.

So far we can represent NEXPTIME (under some natural encoding ν of finite strings as sets):

M non-deterministically accepts w in time $2^{p(|w|)}$
if and only if $(I_w, Accept) \in g(\nu(w)\,/-)$

where I_w is initial configuration for w (as tape tree),
Accept is unique accepting configuration (as tape tree).

Representing Alternation

More is possible: M alternating Turing machine, i.e. states are labelled either $\wedge$ or $\vee$.
An M-configuration c is accepting iff

1. c is labelled $\vee$ and some of c's immediate successor configurations are accepting; or
2. c is labelled $\wedge$ and all of c's immediate successor configurations are accepting.

$$h(x, y\,/-) \mapsto \{c \in f(x\,/-):\ c \text{ is accepted by } M \text{ in exponential time with } \leq \mathrm{rank}(y) \text{ many alternations}\}$$

is SR by safe recursion on y.

Main Result on HF

A natural encoding of finite strings as sets:

$\nu(s\,i) =$ the ordered pair $(i, \nu(s)) = \{\{i\}, \{i, \nu(s)\}\}$

Theorem (B., Buss '11)

Under the above encoding, the SR functions on finite strings are exactly the functions computed by alternating Turing machines running in exponential time with polynomially many alternations.

Remark

L. Berman [*The complexity of logical theories*, TCS, 11 (1980), pp. 71–77]: this complexity class exactly characterizes the complexity of validity in theory of real numbers as an ordered additive group.

Computing SR functions by Turing Machines

Problem with proving converse of Main Theorem is that sizes of sets can get too big to be stored on tape of exponential length!

Thus, instead of dealing with sets directly, we consider the following test: (Fix some well-ordering on HF sets.)

> *Given:* $x \in \mathrm{HF}$, *and sequence* $i_1, \ldots, i_k \in \mathbb{N}$.
> *Does* i_k*-th element of* i_{k-1}*-th element of . . . of* i_1*-th element of* x *exists?*

Claim: This test for a set computed by some SR function applied to sets coding finite strings, and a sequence $i_1, \ldots, i_k \in \mathbb{N}$, can be computed by alternating Turing machines in exponential time with polynomial many alternations.

Functions based on encodings

The natural encoding above: $\nu(s\,i) = (i, \nu(s))$

Any encoding $\nu : \{0,1\}^* \to \mathrm{HF}$ gives rise to class of functions in the following way:
SR set function F defines function $f : \{0,1\}^* \to \{0,1\}^*$ by

$$\begin{array}{ccc} \mathrm{HF} & \xrightarrow{F} & \mathrm{HF} \\ \uparrow \nu & & \uparrow \nu \\ \{0,1\}^* & \xrightarrow{f} & \{0,1\}^* \end{array}$$

Ackermann encoding

Identify $\{0,1\}^*$ with $\mathbb{N}$.

$$\mathrm{Ack}(i_k \ldots i_1) = \{\mathrm{Ack}(j) \colon i_j = 1,\ j = 1, \ldots, k\}$$

Encoding is very shallow: $\mathrm{rank}(\mathrm{Ack}(w)) \approx \log^*(|w|)$.

Resulting class of functions not nice, e.g. the predecessor function is not computable by a dietary function:

Let s be $1\,0^{2_k-1}$ (in binary).
Hence predecessor s' of s is 1^{2_k-1} (in binary),
$\mathrm{rank}(\mathrm{Ack}(s)) = O(k)$, $\mathrm{card}(\mathrm{tc}(\mathrm{Ack}(s))) = O(k)$
but $\mathrm{card}(\mathrm{Ack}(s')) \geq 2_k$

An intermediate encoding:

Define

$$\nu^*(w) \;=\; (\nu(\log(|w|)), \mathrm{Ack}(w))$$

The resulting class of functions are those computable in
time $2^{(\log n)^{O(1)}}$
alternations $\leq (\log n)^{O(1)}$
that is computable in quasi-polytime with poly-logarithmic many alternations.

SR Set Functions and the L-Hierarchy

SR Set Functions on general sets.

Following Jensen, we define

Definition (SR-closure)

SR-closure(A) := least SR-closed $B \supseteq A$
For transitive T, SR(T) := SR-closure($T \cup \{T\}$)

Theorem (Sy Friedman)

For transitive T

$$\mathrm{SR}(T) = L^T_{\mathrm{rank}(T)^\omega}$$

where L^T is the L-hierarchy relativised to T.

Definability Characterisation of SR Set Functions

For any $\vec{x}$ let $\mathrm{TC}(\vec{x})$ be the transitive closure of $\vec{x}$. The function $\vec{x} \mapsto \mathrm{TC}(\vec{x})$ is SR. Define

$$\mathrm{SR}(\vec{x}) := \mathrm{SR}(\mathrm{TC}(\vec{x})) = L^{\mathrm{TC}(\vec{x})}_{\mathrm{rank}(\vec{x})^\omega}$$

$$\mathrm{SR}'_n(\vec{x}) := L^{\mathrm{TC}(\vec{x})}_{\mathrm{rank}(\vec{x})^n} \text{ for finite } n$$

Theorem (Sy Friedman)

Suppose that $f(\vec{x}\,/-)$ is SR. Then for some Σ_1 formula φ and some finite n we have:

$$f(\vec{x}\,/-) = y \quad \textit{iff} \quad \mathrm{SR}'_n(\vec{x}) \vDash \varphi(\vec{x}, y)$$

Conversely, any function so defined is SR.

The SR Hierarchy

Analogue of Jensen's hierarchy:

$\mathrm{SR}_1 := \mathrm{HF}$, the collection of hereditarily finite sets
$\mathrm{SR}_{\alpha+1} := \mathrm{SR}(\mathrm{SR}_\alpha)$ for $\alpha > 0$
$\mathrm{SR}_\lambda := \bigcup_{\alpha<\lambda} \mathrm{SR}_\alpha$ for limit λ

Corollary (Sy Friedman)

For every α, $\mathrm{SR}_{1+\alpha} = L_{\omega(\omega^\alpha)}$.

$L_\omega \subseteq L_{\omega^\omega} \subseteq L_{\omega(\omega^2)} \subseteq L_{\omega(\omega^3)} \ldots$

SR Set Functions on Binary Strings of Length ω

For $\vec{x}$ a finite sequence of binary ω-strings, we have $\mathrm{SR}(\vec{x}) = L_{\omega^\omega}[\vec{x}]$ as $\mathrm{rank}(\vec{x}) < \omega + \omega$.
Thus, the SR functions on ω-strings are characterised by

$$f(\vec{x}\,/-) = y \quad \text{iff} \quad L_{\omega^n}[\vec{x}] \models \varphi(\vec{x}, y)$$

for some Σ_1 formula φ and some finite n.

Corollary

The SR functions on ω-strings coincide with those computable by an infinite-time Turing machine in time ω^n for some finite n (as considered by Deolaliker, Hamkins, Schindler, Welch and others.)

Recent Work by Toshiyasu Arai

Arai weakened our schemes for SR set functions, obtaining his *PC (predicatively computable) set functions.* Recall that we used:
Rudimentary union scheme $f(\vec{x} \,/\, \vec{a}, b) = \bigcup_{z \in b} g(\vec{x} \,/\, \vec{a}, z)$

Arai replaces this by

Null: $(-/\, b) = \emptyset$
Union: $\mathrm{union}(-/\, b) = \bigcup b$
Conditional$_\in$*:* $\mathrm{Cond}_\in(-/\, a, b, c, d) = \begin{cases} a & \text{if } c \in d \\ b & \text{otherwise.} \end{cases}$

plus closure under

Safe Separation Scheme
$f(-/\, \vec{a}, c) = \{b \in c \colon h(-/\, \vec{a}, b) \neq \emptyset\}$

$\big[$implies a more strict union scheme $f(x, \vec{y} \,/\, \vec{a}) = \bigcup_{z \in x} g(z, \vec{y} \,/\, \vec{a})$ $\big]$

Recent Work by Toshiyasu Arai

On HF:

Theorem (Arai)

The PC set functions on finite strings are exactly the polytime functions.

On infinite sets:

Theorem (Arai)

*The PC set functions are exactly the functions Σ_1-definable in $KP^-(\mathcal{D})$ + **($\Sigma_1(\mathcal{D})$-Submodel Rule)** + **($\Sigma_1(\mathcal{D})$-Foundation)**.*

KP^-: *KP* minus foundation
$\mathcal{D}$: "normal" values

Summary

- Safe Recursive Set Functions
  $$= \quad \text{Bellantoni-Cook} \quad + \quad \text{Primitive Set Recursion}$$
- SR Set Functions with natural encoding of finite strings characterise alternating EXPTIME with polynomially many alternations
- SR Set Functions coincide with other proposed notions of polytime on ω-strings (Infinite Time Turing Machines)

Take Away Message:

Safe Recursive Set Functions provide
an adequate notion of feasible computation on infinite sets.

Thanks for listening to my talk

Preprint available at:

`http://www.cs.swan.ac.uk/~csarnold/publ/show-paper.php?27`

References


Toshiyasu Arai. *Predicatively computable functions on sets.*
Tech. rep., arXiv.org, 2012, arXiv:1204.5582v2.


Arnold Beckmann and Andreas Weiermann. *A term rewriting characterization of the polytime functions and related complexity classes.*
Archive for Mathematical Logic 36:11–30, 1996.


Stephen Bellantoni and Stephen Cook. *A new recursion-theoretic characterization of the polytime functions.*
Comput. Complexity, 2(2):97–110, 1992.


Leonard Berman. *The Complexity of Logical Theories.*
Theoretical Computer Science 11:71–77, 1980.


R. Björn Jensen. *The fine structure of the constructible hierarchy.*
Ann. Math. Logic, 4:229–308; erratum, ibid. 4 (1972), 443, 1972.


Ronald B. Jensen and Carol Karp. *Primitive recursive set functions.*
In *Axiomatic Set Thoory (Proc. Sympos. Pure Math., Vol. XIII, Part I, Univ. California, Los Angeles, Calif., 1967)*, pages 143–176. Amer. Math. Soc., Providence, R.I., 1971.


Vladimir Yu. Sazonov. *On Bounded Set Theory.*
In *Logic and Scientific Methods*, pages 85–103. Kluwer Academic Publisher, 1997.