Structural Computational Complexity Theory

Takatoshi Fujisawa

School of Information Science,
Japan Advanced Institute of Science and Technology

February 15, 2000

Keywords: structural computational complexity, polynomial time computable functions, function algebras, recursion schemes, Constable's class $\mathcal{K}$.

1 Background

Before the existence of modern computers, in the 1930's, some mathematical logicians would like to grips to with the notion *effective computation*, in general way that would allow them to distinguish between the computable and the noncomputable, such as A. Turing introduced Turing machine; S. Kleene defined μ-recursive functions based on the study of K. Gödel, J. Herbrand; and A. Church formalized λ-calculus. Because these vastly dissimilar formalisms are all computationally equivalent, the common notion of computability that embody is extremely robust, which is to say that it is invariant under fairly radical perturbations in the models. All these mathematical logicians with their pet systems turned out to be looking at the same thing from different angles. They soon came to the realization that the commonality among all these systems must be the elusive notion of effective computability that they had sought for so long. Church gave voice to this thought, and it has since become known as the *Church's thesis* (or the *Church-Turing thesis*).

However if a function is computable in principle, a function is not always computable feasibly. One has known the problems that is computable in principle and is not feasibly computable. *Computational complexity* deal with the cost or difficulty of computing the computable functions. The field of *structural computational complexity* defines the class corresponding to the difficulty, and study the including relation of some classes.

The aim of this paper is the establishment of the notion of "feasible" computable. In this paper, we use mainly the method of *recursion schemes* and *function algebra* is a smallest class of functions containing certain initial functions and closed under certain operations. And the class of polynomial time computable functions and Constable's class $\mathcal{K}$ are considered as the feasibly computable class.

2 Polynomial time computable functions

The class of polynomial time computable functions (PTIME) was originally defined by using Turing machine. In [5] A.Cobham first isolate the machine independent characterization of polynomial time computable functions as using a certain variant of primitive recursion called *bounded recursion on notation* (BRN). The function f is defined by BRN[1] from g, h_0, h_1, k if

$$\begin{aligned} f(0, \vec{y}) &= g(\vec{y}), \\ f(s_0(x), \vec{y}) &= h_0(x, \vec{y}, f(x, \vec{y})), \text{ if } x \neq 0 \\ f(s_1(x), \vec{y}) &= h_1(x, \vec{y}, f(x, \vec{y})) \end{aligned}$$

provided that $f(x, \vec{y}) \leq k(x, \vec{y})$ for all $x, \vec{y}$. Thus Cobham showed PTIME as the least class of functions which includes certain initial functions and which is closed under composition and BRN. Primitive recursion defines $f(x+1)$ in terms of $f(x)$, so that the computation of $f(x)$ requires approximately $2^{|x|}$ many steps, an exponential number in the length of x. To define smaller complexity classes of functions, Bennet introduced the scheme of *recursion on notation*, which Cobham [5] later used to characterize the polynomial time computable functions. The scheme of *recursion on notation* requires $|x|$ steps to compute $f(x)$.

Although his characterization has yielded a number of applications, unsatisfying aspect of his characterization arises in the bounded recursion on notation. Recently, certain *unbounded* recursion schemes have been introduced to characterize PTIME. In this paper the author picks up the following two results. S. Bellantoni and S. Cook [1] introduced certain unbounded recursion schemes which distinguish between variables as to their position in a function $f(x_1, \ldots, x_n; y_1, \ldots, y_m)$. Variables x_i occurring to the left of the semi-colon are called *normal*, while variables y_i to the right are called *safe*. They defined *safe recursion on notation* (SRN)[2]; the function f is defined by SRN from the functions g, h_0, h_1 if

$$\begin{aligned} f(0, \vec{y}; \vec{a}) &= g(\vec{y}; \vec{a}) \\ f(s_0(x), \vec{y}; \vec{a}) &= h_0(x, \vec{y}; \vec{a}, f(x, \vec{y}; \vec{a})), \text{ provided } x \neq 0 \\ f(s_1(x), \vec{y}; \vec{a}) &= h_1(x, \vec{y}; \vec{a}, f(x, \vec{y}; \vec{a})). \end{aligned}$$

H. Ishihara [6] introduced *full concatenation recursion on notation* (FCRN), inspired by Clote's [2] *concatenation recursion on notation*. Assume that $h_0(x, \vec{y}, z), h_1(x, \vec{y}, z) \leq 1$. The function f is defined by FCRN from g, h_0, h_1 if

$$\begin{aligned} f(0, \vec{y}) &= g(\vec{y}) \\ f(s_0(x), \vec{y}) &= s_{h_0(x, \vec{y}, f(x, \vec{y}))}(f(x, \vec{y})), \quad \text{if } x \neq 0 \\ f(s_1(x), \vec{y}) &= s_{h_1(x, \vec{y}, f(x, \vec{y}))}(f(x, \vec{y})). \end{aligned}$$

The polynomial time hierarchy (PH) is able to be also characterized by these recursion schemes.

[1] $s_0(x) = 2 \cdot x$, $s_1(x) = 2 \cdot x + 1$.

[2] In [1] this scheme is called *predicative recursion on notation*.

3 Constable's class $\mathcal{K}$

R. Constable defined the class $\mathcal{K}$ drawing on polynomial analogy with the Kalmár elementary functions. The class of elementary functions $\mathcal{E}$ is the smallest class containing initial functions; 0, projection functions, $x+1$, $+$, $\dot{-}$[3] and closed under composition, *bounded summation* (BSUM) and *bounded product* (BPROD). Then the function f is defined by BSUM [resp. BPROD] from g if $f(x,\vec{y})$ equals

$$\sum_{i=0}^{x} g(i,\vec{y}) \qquad [\text{resp. } \prod_{i=0}^{x} g(i,\vec{y})].$$

Since $\mathcal{E}$ embodies exponential function, we can not regard $\mathcal{E}$ as feasible class. On the other hand $\mathcal{K}$ is the smallest class of functions containing the initial functions; 0, projection functions, $2\cdot x$, $2\cdot x+1$, $+$, $\dot{-}$, $\times$, $\lfloor x/y \rfloor$ and closed under composition, *sharply bounded summation* (SBSUM) and *sharply bounded product* (SBPROD). Thus the function f is defined by SBSUM [resp. SBPROD] from g if $f(x,\vec{y})$ equals

$$\sum_{i=0}^{|x|} g(i,\vec{y}) \qquad [\text{resp. } \prod_{i=0}^{|x|} g(i,\vec{y})].$$

In [3], P. Clote showed the relation $\mathcal{K}$ and some parallel complexity classes defined by circuit families. Hence we have following,

$$\mathrm{TC}^0 \subseteq \mathcal{K} \subseteq \mathrm{NC}.$$

Where for $k \geq 0$, NC^k is the class of languages accepted by LOGTIME-uniform, polynomial *size* and $O(\log^k n)$ *depth* over the boolean basis, where $\wedge, \vee$ have fan-in 2, and $\mathrm{NC} = \cup_k \mathrm{NC}^k$. And TC^0 is the class of languages in LOGTIME-uniform polynomial *size*, constant *depth* over the boolean basis of threshold gate. Moreover we already know that the machine independent characterization of NC [2] and TC^0 [4].

The class $\mathcal{K}$ is very natural, however the corresponding machine model is unknown. Then following question is suggesting.

"What complexity class corresponds to the class $\mathcal{K}$?"

H.-J. Burtshick has proposed that polynomial size uniform arithmetic circuits could be related to the $\mathcal{K}$. H. Ishihara suggested that the algebra adding initial function $x^{|y|}$ to TC^0 could be involved with $\mathcal{K}$. And Ishihara showed that TC^0 is closed under SBSUM. Then we define the class $\mathcal{T}$ as the class appending initial functions $x^{|y|}$ and $\lfloor x/y \rfloor$ to TC^0 and attempt to show the equivalence of $\mathcal{T}$ and $\mathcal{K}$. By previous discussion if $\mathcal{T}$ is closed under SBPROD then $\mathcal{T} = \mathcal{K}$. Although the author showed that $\mathcal{T}$ contains particular binary coefficient $\binom{|x|}{y}$ and also the particular factorial function of binary length $|x|!$, the author is coming to reluctant conclusion that the question remains as still open problem. It is left as a future work.

[3] $x \dot{-} y$ equals if $x - y \geq 0$ then $x - y$ else 0.

References

[1] S. Bellantoni and S. Cook. A new recursion-theoretic characterization of the polytime functions. *Computational Complexity,* 2:97-110, 1992.

[2] P. Clote. Sequential, machine-independent characterizations of the parallel complexity classes $ALOGTIME$,AC^k,NC^k and NC. In P. J. Scott, S. R. Buss, editor, *Feasible Mathematics,* pages 49-70, Birkhäuser, 1990.

[3] P. Clote. A Note on the Relation Between Polynomial Time Functionals and Constable's Class $\mathcal{K}$. In Kleine-Büning, editor, *Computer Science Logic.* Springer Lecture Notes in Computer Science, 1996. Result presented at CSL in Paderborn, 1995. To appear.

[4] P. Clote and G. Takeuti. First order bounded arithmetic and small boolean circuit complexity classes. In P. Clote and J. Remmel, editors, *Feasible Mathematics II,* pages 154-218, Birkhäuser, 1995.

[5] A. Cobham. The intrinsic computational difficulty of functions. In Y. Bar-Hillel,editor, *Logic, Methodology and Philosophy of Science II,* pages 24-30, North-Holland, 1965.

[6] H. Ishihara. Function algebraic characterizations of the polytime functions. *Computational Complexity,* to appear, 1998.