Composition is the only independent rule in Kleene's model of partial recursive functions

Gheorghe M. Ştefan

http://users.dcae.pub.ro/~gstefan/

Abstract

Out of the three rules defined by Stephen Kleene in its concept of partial recursive functions [8] only the composition rule is independent. We prove in this short note the other two rules are repeated applications of specific compositions.

Preamble

When, around 600 BC, the semi-mythical seer Epimenides of Cnossos, so an inhabitant of Crete, reportedly stated that "All Cretans are liars", in the Occidental culture all the logical minds get in trouble for more than two and half millenia, until, after a rigorous reformulation of the problem by David Hilbert (1862 – 1943) [6] [7], the logician Kurt Friedrich Gödel (1906 – 1978) proved that the truth of such a sentence is not decidable [5]. Then, after only five years, independently, but almost "synchronously", four mathematicians – Alonzo Church (1903 – 1995) [3], Stephen Cole Kleene (1909 – 1994) [8], Emil Leon Post (1897 – 1954) [9], and Alan Mathison Turing (1912 – 1954) [10] – published their mathematical version for Gödel's incompleteness theorem. All the four versions proved to be equivalent, but Turing's solution was first considered as the source of an engineering solution for computation. Thus, the most important negative result in the history of logic – the Gödel's incompletness theorem – triggered the start of the computing era. Now, after more than half a century, it is time for another approach to take the lead. To substantiate the parallel computation, Kleene's solution seems to fit better.

Kleene's Model of Partial Recursive Functions

Definition 1 *Let be the positive integers $x, y, i \in \mathbb{N}$ and the vector $\vec{x} = \langle x_0, x_1, \ldots, x_{n-1} \rangle \in \mathbb{N}^n$. Any partial recursive function $f : \mathbb{N}^n \to \mathbb{N}$ can be computed using three **initial functions**:*

- $ZERO(x) = 0$: *the variable x takes the value* **zero**
- $INC(x) = x + 1$: **increment***s the variable $x \in \mathbb{N}$*
- $SEL(i, \vec{x}) = x_i$: *i* **select***s the value of x_i from the vector of positive integers $\vec{x}$*

*and the application of the following three **rules**:*

- **Composition:** $f(\vec{x}) = g(h_0(\vec{x}), \ldots, h_{p-1}(\vec{x}))$, *where: $f : \mathbb{N}^n \to \mathbb{N}$ is a total function if $g : \mathbb{N}^p \to \mathbb{N}$ and $h_i : \mathbb{N}^n \to \mathbb{N}$, for $i = 0, 1, \ldots p-1$, are total functions*
- **Primitive recursion:** $f(\vec{x}, y) = g(\vec{x}, f(\vec{x}, (y-1)))$, *with $f(\vec{x}, 0) = h(\vec{x})$ where: $f : \mathbb{N}^{n+1} \to \mathbb{N}$ is a total function if $g : \mathbb{N}^{n+1} \to \mathbb{N}$ and $h : \mathbb{N}^n \to \mathbb{N}$ are total functions.*
- **Minimization:** $f(\vec{x}) = \mu_y[g(\vec{x}, y) = 0]$, *which means: the value of the function $f : \mathbb{N}^n \to \mathbb{N}$ is the smallest y, **if any**, for which the function $g : \mathbb{N}^{n+1} \to \mathbb{N}$ takes the value $g(\vec{x}, y) = 0$.*

◇

Kleene's model looks like a good candidate for a mathematical model for parallel computing as the Turing's model was for the mono-core computation. In this respect, the composition seems to be a natural embodiment of a many-core abstract model for the parallel computing engine. The following conjecture has a big chance to become a theorem:

Conjecture 1 *The composition rule, implemented as a two level structure (see Figure 1):*

- *the* **MAP** *level: a linear array of circuits, one for each $h_i(\vec{x})$ function*
- *the* **REDUCE** *level: a* log*-depth tree-like network of circuits for $g(h_0(\vec{x}), \ldots, h_{p-1}(\vec{x}))$*

where the functions $h_i(\vec{x})$ and the function $g(h_0(\vec{x}), \ldots, h_{p-1}(\vec{x}))$ are initial functions or hierarchic compositions of initial functions, computes any functions $f : \mathbb{N}^n \to \mathbb{N}$.

◇

Two kinds of parallelism are emphasized by the composition rule:

- a n-degree of synchronic parallelism between the computation performed in the circuits of the MAP level

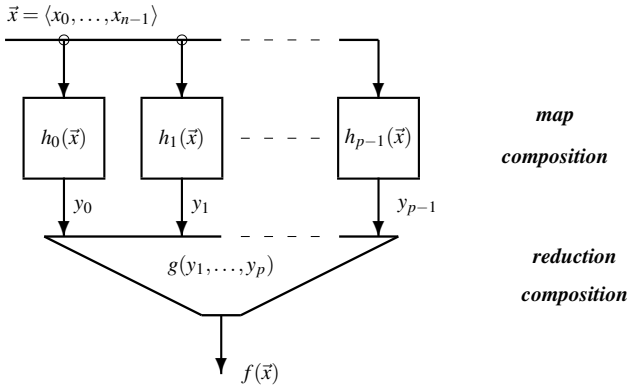

Figure 1: **The circuit version of composition**. It is a two-layer construct: the parallel expanded *map* layer serially connected with the *reduction* layer.

- a 2-degree of diachronic (pipeline) parallelism between the computation on the map level and the computation of the REDUCE level

In the next sections will be proved that, for the other two rules, specific compositions can be used, so as we are in the position to conclude that the computation model proposed by Kleene leads to implementations involving only compositions for which the previous conjecture applies.

Preliminary Definitions

Definition 2 *The reduction-less composition or* map composition*, MC, is the particular composition $f : \mathbb{N}^n \to \mathbb{N}^p$ where:*

$$f(\vec{x}) = f(x_0, \ldots x_{n-1}) = \langle h_0(\vec{x}), \ldots h_{p-1}(\vec{x}) \rangle = \langle y_0, \ldots y_{p-1} \rangle = \vec{y}$$

$h_i : \mathbb{N}^n \to \mathbb{N}$, and $g(\vec{y}) = \vec{y}$ is the identity function, for $i = 0, \ldots, p-1$.

◇

Definition 3 *The map-less composition or* reduction composition*, RC, is the particular composition $f : \mathbb{N}^n \to \mathbb{N}$ where:*

$$f(\vec{x}) = g(\vec{x})$$

with $y_i = h_i(\vec{x}) = SEL(i, \vec{x}) = x_i$, for $i = 0, \ldots, p-1$ and $n = p$.

◇

According to the previous two definitions, the composition rule can be considered as having a ***map-reduce*** structure (Figure 1), where a MC is serially connected with a RC. The two functional level can have associated the physical implementation with the h_i functions and the

g function embodied in various forms, starting from combinational circuits and reaching the complexity and competence of a processor, even a computer.

Definition 4 *The RC function* $redOR : \mathbb{N}^n \to \mathbb{N}$ *is*

$$redOR(\vec{x}) = x_0|x_1|\ldots|x_{n-1}$$

where: $|$ *denote the bitwise OR logical function.*

◇

Definition 5 *For* $\vec{x} = \langle x_0, x_1, \ldots, x_i \ldots \rangle$*, let us consider the right-chained MC*

$$rightMC(x, \vec{x}) = H'(x, \vec{z}) \circ H(\vec{x})$$

composed by the following two MCs:

$$H(\vec{x}) = \langle h_0(\vec{x}), h_1(\vec{x}), \ldots, h_i(\vec{x}) \ldots \rangle = \langle z_0, z_1, \ldots, z_i \ldots \rangle$$

$$H'(x, \vec{z}) = \langle h'_0(x, SEL(0, \vec{z})), h'_1(SEL(0, \vec{z}), SEL(1, \vec{z})), \ldots, h'_i(SEL(i-1, \vec{z}), SEL(i, \vec{z})), \ldots \rangle$$

◇

Thus, *rightMC*s could be associated to serial connections between the cells performing the functions $h_i \circ h'_i$. Theoretically, MAP section is right unlimited. Results the circuit from Figure 2. Similarly, a left serial connection can be defined.

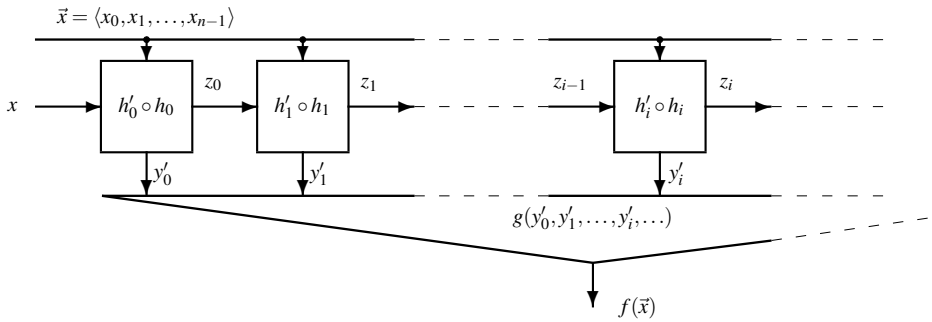

Figure 2: **The circuit associated to the composition rule expanded with a rightSHIFT serial connections.**

Definition 6 *By definition,* $rightSHIFT(x, \vec{x})$ *is a* $rightMC(x, \vec{x})$ *for* $\vec{x} = \langle x_0, x_1, \ldots, x_i \ldots \rangle$*,* $h_i(\vec{x}) = SEL(i, \vec{x})$*,* $h'_0 = x$*, and* $h'_i = SEL(i-1, \vec{z})$*, for* $i = 0, 1, \ldots, i, \ldots$*, such that:*

$$rightSHIFT(x, \vec{x}) = \langle x, x_0, x_1, \ldots, x_i \ldots \rangle$$

◇

Definition 7 *We define* $prefixOR(\vec{b})$ *as a scan circuit for the OR prefix function [1], so it receiving a binary input vector*

$$\vec{b} = \langle b_0, b_1, \ldots, b_i, \ldots \rangle$$

returns

$$prefixOR(\vec{b}) = \langle b_0, b_0|b_1, b_0|b_1|b_2, \ldots, OR_0^i b_j, \ldots \rangle$$

◇

Definition 8 *The MC function* $scanFIRST : \{0,1\}^n \to \{0,1\}^n$ *is:*

$$scanFIRST(\vec{b}) = prefixOR(\vec{b}) \& \ (rigthSHIFT(prefixOR(\vec{b})))$$

where: $\vec{b} = \langle b_0, b_1, \ldots, b_{n-1} \rangle$ *is a Boolean sequence.*

◇

The $scanFIRST(\vec{b})$ function identifies, ***if any***, the first occurrence of 1 in a the Boolean sequence $\vec{b}$.

Primitive Recursion Computed as a Sequence of Compositions

Theorem 1 *The primitive recursive rule is reducible to repeated applications of specific compositions.*

◇

Proof 1 *The primitive recursion rule could be applied using its iteratively expanded form:*

$$f(\vec{x}, y) = g(\vec{x}, f(\vec{x}, y-1)) = \ldots = \underbrace{g(\vec{x}, g(\vec{x}, g(\vec{x}, \ldots g}_{(y-1)\ times}(\vec{x}, f(\vec{x}, 1)) \ldots))) =$$

$$= \underbrace{g(\vec{x}, g(x, g(\vec{x}, \ldots g}_{y\ times}(\vec{x}, f(x, 0)) \ldots))) = \underbrace{g(\vec{x}, g(\vec{x}, g(\vec{x}, \ldots g}_{y\ times}(\vec{x}, h(x)) \ldots)))$$

Let be, in Figure 3, the specific instantiation of the righMC function (see Definition 5). It computes iteratively, starting in the first stage, P_0*, with the function* $f(\vec{x}, 0) = h(\vec{x})$*, the values* $f(\vec{x}, i)$ *for* $i = 0, 1, \ldots$*. In each stage the predicate* $(y = i)$ *is computed. The functions* P_i*, for* $i = 1, 2, \ldots$*, takes form* P_{i-1} *the value of* $f(\vec{x}, i-1)$ *and computes* $f(\vec{x}, i)$*. The redOR function takes from* P_i *its arguments as* $(y = i)\ ?f(\vec{x}, i) : 0$ *for* $i = 0, 1, 2, \ldots$*. Because for only one i the predicate* $y = i$ *takes the value 1, the function redOR returns the value of* $f(\vec{x}, y)$*.*

Thus, for primitive recursion we need to compose two compositions, rightSHIFT and redOR.

◇

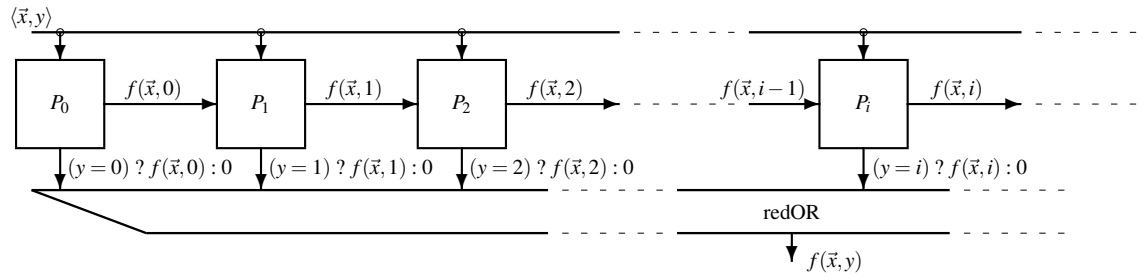

Figure 3: **The *rightMC* & *redOR* circuit version for the partial recursive rule**.

Figure 3 presents the circuit version of the function obtained by composing a specific *rightMC* function with the *redOR* function. The two stage computation just described, as a structure indefinitely extensible to the right, is a theoretical model because the index i takes values no matter how large, similar with the "infinite" tape of Turing Machine. But, it is very important that the algorithmic complexity of the description is in $O(1)$, because the functions P_i, defining *rightMC*, and *redOR* have constant size descriptions.

Minimization Computed as a Sequence of Compositions

Theorem 2 *The minimization (least-search) rule is reducible to repeated applications of specific compositions.*

◇

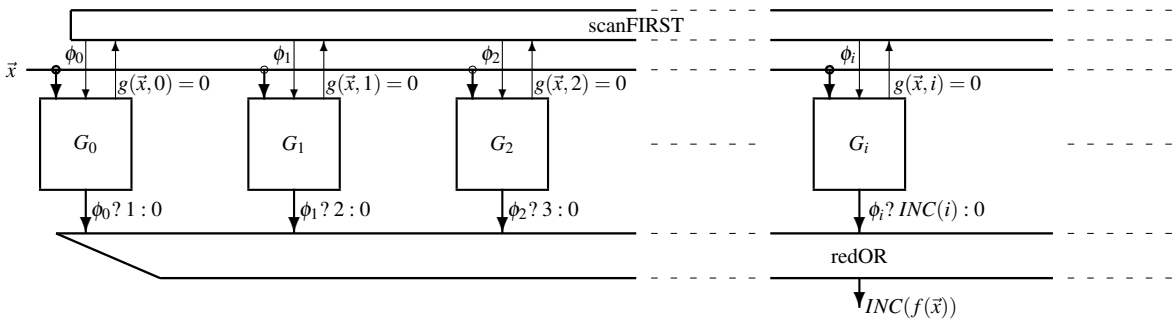

Figure 4: **The circuit structure for the minimization rule**.

Proof 2 *The minimization (least-search) rule computes the value of $f(\vec{x})$ as the smallest y,* **if any***, for which $g(\vec{x},y)=0$. The functional structure to which we will refer is represented in Figure 4.*

The scanFIRST *MC receives from the MAP section the vector of predicates*

$$\langle (g(\vec{x},0)=0),(g(\vec{x},1)=0),\ldots,(g(\vec{x},i)=0),\ldots\rangle$$

and returns the Boolean vector

$$\vec{\phi}=\langle \phi_0,\phi_1,\ldots,\phi_i,\ldots\rangle$$

Thus, scanFIRST *MC points, with $\phi_k=1$, to the first cell,* ***if any****, which provided the predicate $(g(\vec{x},i)=0)=1$. The MAP section uses the vector $\vec{\phi}$ to generate the vector:*

$$\langle (\phi_0\ ?\ 1:0),(\phi_1\ ?\ 2:0),\ldots,(\phi_i\ ?\ INC(i):0),\ldots\rangle$$

as input for the reduction section redOR*. Thus, the output of the reduction section takes the value $INC(f(\vec{x}))$, because the value 0 is reserved to indicate that the function is not defined for the value $\vec{x}$ applied on the input.*

◇

The computation just described is only a theoretical model, because the index i has an indefinitely large value. But, the size of the algorithmic description remains $O(1)$.

Partial Recursion Means Composition Only

Kleene's approach defines, besides the composition rule, the other two rules, ordinary (primitive) recursion and minimization (least-search), only for providing the means for classifying the recursive functions (to emphasize in the class of recursive functions the partial recursive functions and primitive recursive functions). Therefor, to define the computation the next corollary makes the necessary and sufficient delimitation.

Corollary 1 *Any computation defined in Definition 1 can be done, according to Theorem 1 and Theorem 2, using the initial functions and the repeated application of the composition rule.*

◇

The primitive recursion is differentiated from the partial recursion by the main fact that it does not require the scan section. It is only a straight forward sequence of compositions. For the partial recursive rule, the additional scan section interferes with the sequence of compositions in order to validate intermediate results. Thus, an actual abstract model based on the Kleene's

computational model must provide also a sequencing mechanism. This means at least to provide programmability at the cells level and an external control.

Corollary 1 can be used to define an abstract model for parallel computation. The resulting structure is able to support the heterogeneous solution for advanced parallel computation. A Church inspired lambda architecture [3] can be defined for the complex control, while a Kleene inspired architecture could be used for solving problems raised by the intense computation. The operating systems problems can remain to be handled by Turing/Post [10] [9] inspired engines.

Important note: the composition rule is defined for a finite p, while the various forms of the composition rule used to define primitive recursion and minimization are designed for a large, mathematically speaking, infinite p. Moving to an abstract model for computing will be a solution that removes this infinity.

References

[1] M. Antonescu, G. M. Stefan: Multi-Function Scan Circuit, *Proceedings of the 43nd International Semiconductor Conference CAS 2020*, October 2020, Sinaia, Romania, pp 123-126.

[2] V. E. Beneš: Optimal Rearrangeable Multistage Connecting Networks, *Bell System Tech. J.*, **43**(4):1646-1656, 1964.

[3] A. Church, An unsolvable problem of elementary number theory, *American Journal of Mathematics* **58**(2):345–363, 1936. Republished in [4].

[4] M. Davis, *The Undecidable. Basic Papers on Undecidable Propositions, Unsolvable Problems and Computable Functions*, Dover Publications, Inc., Mineola, New-York, 2004.

[5] K. Gödel, Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme, I (On Formally Undecidable Propositions of Principia Mathematica and Related Systems), *Monatshefte für Mathematik und Physik* **38**:173–198, 1931. Republished in [4].

[6] D. Hilbert, *Mathematical Problems. Lecture Delivered Before the International Congress of Mathematicians at Paris in 1900*. Available at: `https://www.ams.org/journals/bull/1902-08-10/S0002-9904-1902-00923-3/S0002-9904-1902-00923-3.pdf`

[7] D. Hilbert and W. Ackermann, *Grundzüge der theoretischen Logik* (Principles of Mathematical Logic). Springer-Verlag 1928.

[8] S. Kleene, General recursive functions of natural numbers, *Mathematische Annalen* **112**(5):727–742, 1936. Republished in [4].

[9] E. Post, Finite Combinatory Processes – Formulation 1, *Journal of Symbolic Logic* **1**(3):103–105, 1936. Republished in [4].

[10] A. Turing, On computable numbers with an application to the Eintscheidungsproblem, *Proceedings of the London Mathematical Society*, **42**(1):230–256, 1936 and a corection in **43**(6):544–546, 1937. Republished in [4].