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

Gheorghe M. Ştefan

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

Out of the three rules defined by Stephen Kleene in its concept of partial recursive functions (*3*) only the composition rule is independent. The other two we prove is this short note are repeated applications of specific compositions.

1 Kleene's Model of Partial Recursive Functions

Definition 1 *Let be the positive integers* $x,y,i \in \mathbf{N}$ *and the sequence* $X = \langle x_0, x_1, \ldots, x_{n-1} \rangle \in \mathbf{N}^n$. *Any partial recursive function* $f : \mathbf{N}^n \to \mathbf{N}$ *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* $x \in \mathbf{N}$
- $SEL(i,X) = x_i$: i **select***s the value of* x_i *from the sequence of positive integers X*

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

- **Composition:** $f(X) = g(h_1(X), \ldots, h_p(X))$, *where:* $f : \mathbf{N}^n \to \mathbf{N}$ *is a total function if* $g : \mathbf{N}^p \to \mathbf{N}$ *and* $h_i : \mathbf{N}^n \to \mathbf{N}$, *for* $i = 1, \ldots p$, *are total functions*
- **Primitive recursion:** $f(X,y) = g(X, f(X,(y-1)))$, *with* $f(X,0) = h(X)$ *where:* $f : \mathbf{N}^{n+1} \to \mathbf{N}$ *is a total function if* $g : \mathbf{N}^{n+1} \to \mathbf{N}$ *and* $h : \mathbf{N}^n \to \mathbf{N}$ *are total functions.*
- **Minimization:** $f(x) = \mu y[g(x,y) = 0]$, *which means: the value of the function* $f : \mathbf{N} \to \mathbf{N}$ *is the smallest y,* ***if any****, for which the function* $g : \mathbf{N}^2 \to \mathbf{N}$ *takes the value* $g(x,y) = 0$.

$\diamond$

2 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:*

$$f(X) = f(x_0, \ldots x_{n-1}) = \langle h_1(X), \ldots h_p(X) \rangle = \langle y_1, \ldots y_p \rangle$$

$h_i : \mathbb{N}^n \to \mathbb{N}$*, and* $g(y_1, \ldots, y_p) = \langle y_1, \ldots, y_p \rangle$ *is the identity function, for* $i = 1, \ldots, p$*.*

◇

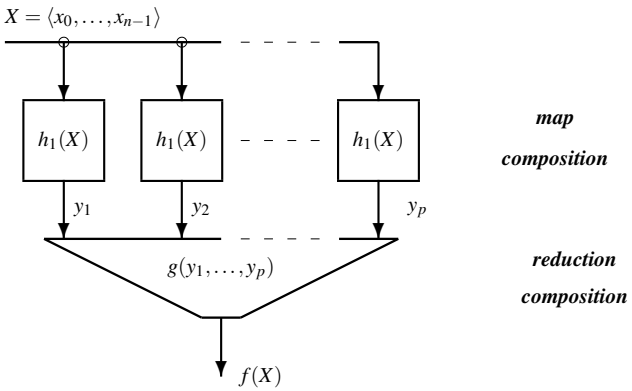

Figure 1: **The circuit version of composition**. It is a two-layer construct: the parallel expanded *map* layer serially connected with the *reduction* layer.

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

$$f(X) = f(x_0, \ldots x_{n-1}) = g(x_1, \ldots x_p)$$

with $y_i = h_i(X) = SEL(i-1, X) = x_{i-1}$*, for* $i = 1, \ldots, p$ *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 function $C_i : \mathbb{N}^{i\times n} \to \mathbb{N}^{(i+1)\times n}$ is a MC defined as:*

$$C_i(Y) = \langle Y, P_i(X_i)\rangle = \langle X_1, \ldots, X_i, P_i(X_i)\rangle$$

where: $Y = \langle X_1, \ldots, X_i\rangle$, the argument, is a sequence of sequences with $X_j \in \mathbb{N}^n$, for $j = 1, \ldots, i$, while $h_j(Y) = SEL(j,Y) = X_j$ for $j = 1, 2, \ldots, i$ and $h_{i+1}(Y) = P_i(SEL(i,Y)) = P_i(X_i)$.

◇

The C function adds, at the end of the sequence X, a new element computed from the last element of X.

Definition 5 *The k-time application of C_i (see Figure 2a), starting from $C_1(\langle X\rangle) = \langle X, P_1(X)\rangle$, where the argument $\langle X\rangle$ is one component sequence of n scalars, defines the* multi-output pipeline*, MOP:*

$$MOP(X) = \langle X, P_1(X), \ldots, P_k(P_{k-1}(\ldots, (P_1(X)\ldots))), \ldots\rangle$$

The resulting structure, with one sequence, $Y = \langle X\rangle$, as argument and as many as necessary computed values, $P_k(P_{k-1}(\ldots, (P_1(X)\ldots)))$, is represented in Figure 2b.

◇

The function $MOP(X)$ is a total function if the functions P_i are total functions, since it is computed using only the repeated application of the composition C_i. For the theoretical model, k is not limited to a specific value. By defining this sequence of MCs, a left to right serial connection between cells is added to the general form of the MC structure.

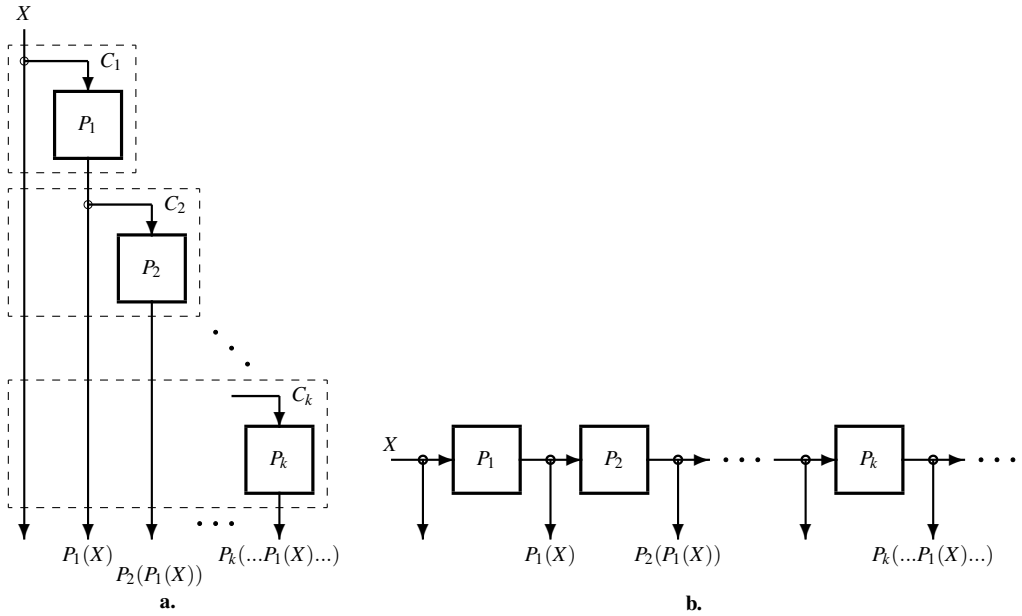

Figure 2: **The multi-output pipeline structure (MOP)**. **a**. The explicit application of C_i. **b**. The resulting MOP circuit structure.

Definition 6 *The function* $PS : \{\{0,1\} \times \mathbb{N}\}^n \to \mathbb{N}$*, called* predicated selection*, is*

$$PS(S) = ADD(V(SEL(1,S)),\ldots,V(SEL(i,S)),\ldots)$$

where: $S = \langle\langle p_1, s_1\rangle, \langle p_2, s_2\rangle, \ldots\rangle$*, takes a sequence of pairs* $\langle predicate, scalar\rangle$*, and returns a scalar. The function ADD adds the scalars validated by the function* $V(\langle p, s\rangle) = p\ ?\ s\ :\ 0$*.*

◇

V is the function *mapped* on the first level of the composition *PS*, while ADD is the *reduction* function associated to the same composition. When no more than one p_i takes the value 1, the function is used to select a scalar from the vector $S = \langle s_1, s_2, \ldots\rangle$.

Definition 7 *The prefixOR function is defined as follows:*

$$prefixOR(x_0, x_1, \ldots, x_i, \ldots) = \ldots OR_i(\ldots OR_2(OR_1(OR_0(x_0, x_1, \ldots, x_i, \ldots)))\ldots)\ldots$$

with:

$$OR_i(x_0, x_1, \ldots, x_i, \ldots) = \langle x_1, \ldots, (x_i | x_{i+1}), x_{i+2}, \ldots \rangle$$

where | stands for the Boolean function OR.

◇

Definition 8 *The ScanFIRST(X) function, , where $X = \langle x_0, x_1, \ldots \rangle$ is defined as follows:*

$$ScanFIRST(X) = prefixOR(X) bwAND(bwNOT(\{0, prefixOR(X)\})$$

where bwAND is the bit-wise logic function AND and bwNOT is the bit-wise logic negation.

◇

The ScanFIRST(X) function identifies the first occurrence of 1 in a Boolean sequence.

3 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:

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

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

Let be, in Figure 3, the specific instantiation of the MOP function (see Definition 5). It computes iteratively, starting in the first stage with the function $f(x,0) = h(x)$, the values $f(x,i)$ for $i = 0, 1, \ldots$. In each stage the predicate $i == y$ is provided. The PS function takes from the

MOP its arguments as pairs of $D_i = \langle predicate_i, value_i \rangle = \langle (i == y), f(x,i) \rangle$. Because only one D_i has $predicate_i = 1$, the function PS returns the result.

Formally, the functions P_i return a pair as follows:

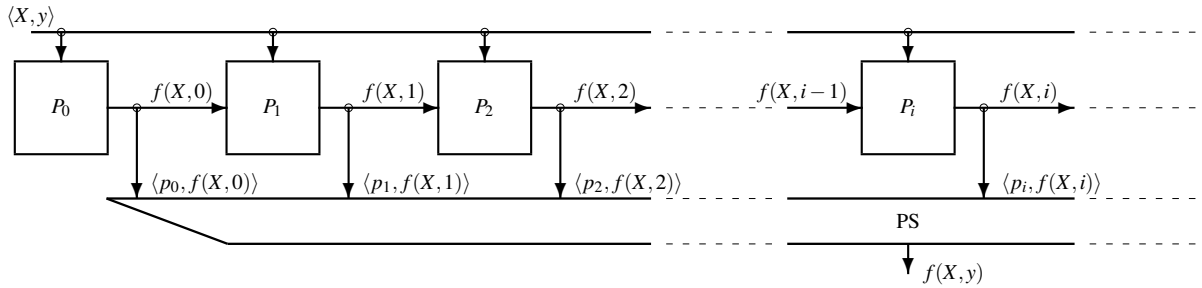

Figure 3: **The MOP structure for the partial recursive rule**.

$$P_i(X, y, p_{i-1}, f(X, i-1) = \langle p_i, f(X,i) \rangle$$

where: $p_i = (i == y)$ and

$$P_0(X, y, -, -) = \langle p_0, h(X,i) \rangle$$

for $i = 0, 1, 2, \ldots$.

From the pairs computed by the MOP function the reduction function PS selects to output the value of $f(X,y)$ because is accompanied by the predicate p_y.

Thus, for primitive recursion we need to compose two compositions, MOP and PS.

◇

Figure 3 presents the circuit version of the function obtained by composing a specific MOP function with the PS 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, MOP and PS have constant size descriptions.

4 Minimalization Computed as a Sequence of Compositions

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

◇

Proof 2 The minimization (least-search) rule computes the value of $f(x)$ as the smallest y, **if any**, for which $g(x,y) = 0$.

Let be, a specific structure from Figure 4. Each cell on the map level computes the pair $\langle predicate, value \rangle$:

$$G_i(x, \phi_i) = \langle (g(x,i) == 0), (\phi_i \; ? \; (i+1) : 0) \rangle$$

The reduction level computes RedOR selecting to the output the value $i+1$ for $\phi_i = 1$, if any. The ScanFIRST loop points to the first cell, if any, which provided the predicate $(g(x,i) == 0) = 1$. If for $x = a$ the output is 0, then the function is not defined for $x = a$, else the output has the value $f(x)+1$.

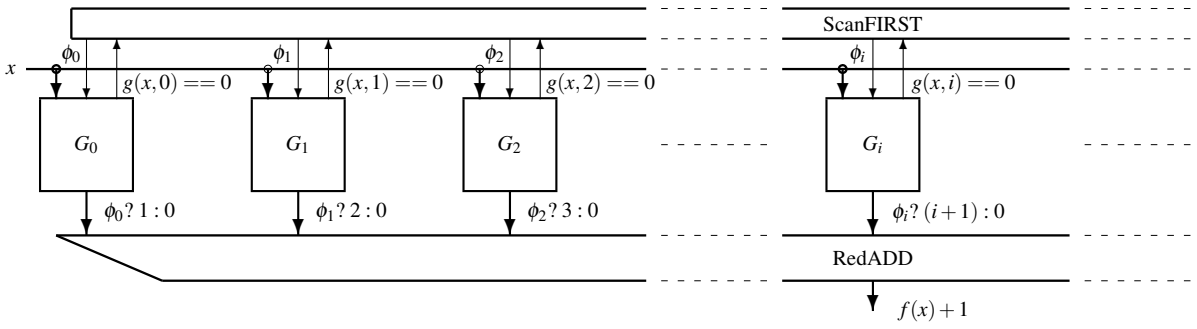

Figure 4: **The structure for the minimalization rule**.

◇

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 $O(1)$.

5 Partial Recursion Means Composition Only

Kleene's approach defines the other two rules, ordinary (primitive) recursion and minimization (least-search), only for helping in establishing the taxonomy of the recursive functions. For defining 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.*

◇

We can conclude that the computation of a partial recursive function is reducible to the multiple application of the composition rule. The primitive recursion and minimalization are emphasized in the definition of the partial recursive functions only to provide the means for classifying the recursive functions (to emphasize in the class of recursive functions the partial recursive functions and primitive recursive functions).

The primitive recursion is differentiated from the partial recursion by the main fact that it does not require the scan loop. It is only a straight forward sequence of compositions. For the partial recursive rule the additional loop 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 hybrid solution for advanced parallel computation. A Church

inspired lambda architecture (*1*) can be defined for 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 (*5*) (*4*) inspired engines.

References

1. A. Church, An unsolvable problem of elementary number theory, *American Journal of Mathematics* **58**, 2 (1936). Republished in (*2*).

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

3. S. Kleene, General recursive functions of natural numbers, *Mathematische Annalen* **112**, 5 (1936). Republished in (*2*).

4. E. Post, Finite Combinatory Processes – Formulation 1, *Journal of Symbolic Logic* **1**, 3 (1936). Republished in (*2*).

5. A. Turing, On computable numbers with an application to the eintscheidungsproblem, *Proceedings of the London Mathematical Society*, **42** and **43**, (1936-7). Republished in (*2*).