In Situ Computations of Mappings.

Abstract. We prove that for every set S and every $k > 0$ and every mapping $E : S^k \mapsto S^k$, every transformation $X \mapsto E(X)$ is computable by a sequence of sucessive modifications of the components of X according to its current values.

1. Introduction-Motivations.

In mathematics, it is usually sufficient for effectiveness to give a formal definition of a mapping E. In computer science, one has to give also a program for the computation of E. Moreover, when the size of the elements is larger than the size of objects the machine can deal with, the program will have to decompose the computation in elementary operations on elementary objects. For instance, a 64 bits processor can only perform operations on $64bits$ and a transformation of an image made of 1000×1000 pixels of 64 bits must be decomposed in successive operations on $64bits$. This fact leads to a well known problem : during the transformation of a vector X by a mapping E, the initial data of X will change and at some step, the program may not be able to complete the whole computation. For example, in order to exchange the contents of two 64 bits registers A, B, if the program begins to do $A := B$, half of the job is done, but since the initial value of A is crushed, this exchange is not possible anymore. The usual solution is to make a copy of the initial data. For instance, this exchange will be computed by $A' := A; A := B; B := A'$. Nevertheless, this solution by copies can generate some memory errors when the structures are too large or at least decrease the performances of the computations. For instance, an operation in a micro-processor involving several registers will have to make copies of these registers in the cache memory or in RAM and refer to these out-memories in order to perform the successive operations.

In this paper, we are interrested in another solution : for computing a state transition $X := E(X)$, just try to do that with X only. For the previous exchange of two registers, that is possible by the following program : $A := A \oplus B; B := A \oplus B; A := A \oplus B$ where $\oplus$ is the addition of respective bits. We are going to show that this kind of "in situ computations" is possible in a very general way.

The second motivation of this paper concerns one-to-one mappings. Given a transition $X := E(X)$ where E is one-to-one, we expect to define such an "in situ computation" that can be easily reversed in an "in situ computation" of the inverse transition $X := E^{-1}(X)$. Hence, if one can compute an image, one can also compute the inverse image with a similar computational complexity.

2. From infinite to two.

We will begin with the easy case (under the axiom of choice). Les S be an infinite set and E be a mapping $S^k \mapsto S^k$ for $k > 0$. How to compute the

transition $X := E(X)$ given X. Of course, this infinite case seems far from computer science since computers can only work with finite data structures. However, we will consider this case for exhaustivity and because it appears to be the easiest : in infinite sets, one can preserve informations by coding lists of elements in one element. More precisely, by a result of Tarski (1924), the axiom of choice is equivalent to the existence of a pairing function on S, that is to say, an injective mapping $\phi_2 : S^2 \mapsto S$. Hence, for every $k > 0$, there exists also an injective mapping $\Phi : S^k \mapsto S' \subseteq S$. For every $i = 1..k$, let π_i be the mapping $S' \mapsto S$ such that

$$\pi_i(\Phi(x_1, \ldots, x_i, \ldots, x_k)) = x_i$$

Now, the program for computing transitions $X := E(X)$ for a given vector $X = (x_1, x_2, \ldots, x_k) \in S^k$ just consists in $k + 1$ steps :

$$\begin{aligned}
&x_1 := \Phi(x_1, x_2, \ldots, x_k)\\
&x_2 := E_2(\pi_1(x_1), \pi_2(x_1), \ldots, \pi_k(x_1))\\
&x_3 := E_3(\pi_1(x_1), \pi_2(x_1), \ldots, \pi_k(x_1))\\
&\ldots\\
&x_k := E_k(\pi_1(x_1), \pi_2(x_1), \ldots, \pi_k(x_1))\\
&x_1 := E_1(\pi_1(x_1), \pi_2(x_1), \ldots, \pi_k(x_1))
\end{aligned}$$

However, one has to assume the computability of these mappings $\Phi, \pi_1, \ldots, \pi_k$. But what does it mean exactly for computers ? For example, for $S = \mathbb{N}$, one can define $\Phi : \mathbb{N}^k \mapsto \mathbb{N}^*$

$$\Phi(x_1, x_2, \ldots, x_k) = 2^{x_1}.3^{x_2} \ldots p_k^{x_k}$$

where p_i is the i-th prime number and define for every $n \in \mathbb{N}^*$

$$\pi_i(n) = k$$

where k is the greatest integer such that p_i^k divides n.

Now assume that S is a finite set. If $|S| \leq 1$, the problem becomes obvious. For $|S| \geq 3$, we are going to reduce this case to the case $S = \{0, 1\}$.
So, given an input vector X In this paper we investigate some way of computing the transformation $X := E(X)$ for mappings $E : S^n \mapsto S^n$ that we call *mappings on* S. In computer science, a data in S^n can be very large and the assignment $X := E(X)$ seems to necessitate the preservation of the whole or a part of the initial value of X. We present here a way to only use the memory of the input data X in order to perform this computation. For example, consider the bijective mapping E on $\{0, 1\}^2$ that maps (a, b) to (b, a). This mapping is linear and bijective. A classical way to compute E in a program is to use a new temporary variable c and compute :

$$c := a, a := b, b := c$$

However, it is known that c is not necessary and E can also be computed with the program :

$$a := a + b, b := a + b, a := a + b$$

In [1], that possibility of computing with no extra variables is proved for any kind of mappings on $\{0,1\}^n$ for any n. In [2], we proved that three types of assignments are sufficient to perform this kind of computations. In [3], we proved that the length of sequential computations is bounded by n^2. In [4], we proved that any linear mapping on $\{0,1\}^n$ is computed with $2n-1$ linear assignments of its boolean components.

In this paper, we improve these previous results. First, we are going to show that for every $n > 0$, every bijective mapping E on $\{0,1\}^n$ is computed by a sequence of $2n-1$ assignments of its components.

Second, we will deduce that for every $n > 0$, every mapping E on $\{0,1\}^n$ is computed by a sequence of $5n-4$ assignments of its components. The idea is to decompose the mapping E in $F \circ P \circ G$ where F and G are bijective and P is a particular mapping that can be computed in just n steps.

2. The case of bijective mappings.

Proposition (bijective). *Let E be a bijective mapping on $\{0,1\}^n$ for $n > 0$. There exists a sequence $f_n, \ldots, f_3, f_2, f_1, g_2, g_3, \ldots, g_n$ of mappings from $\{0,1\}^n$ to $\{0,1\}$ such that the program :*

$$\begin{aligned}
&x_n := f_n(x_n, \ldots, x_1)\\
&\ldots\ldots\\
&x_3 := f_3(x_n, \ldots, x_1)\\
&x_2 := f_2(x_n, \ldots, x_1)\\
&x_1 := f_1(x_n, \ldots, x_1)\\
&x_2 := g_2(x_n, \ldots, x_1)\\
&x_3 := g_3(x_n, \ldots, x_1)\\
&\ldots\ldots\\
&x_n := g_n(x_n, \ldots, x_1)
\end{aligned}$$

computes the assignment $X := E(X)$ for every $X = (x_n, \ldots, x_1)$ in $\{0,1\}^n$.

Proof. By induction on n. For $n = 1$, it is obvious since $X := E(X)$ is computed by the program $x_1 := f_1(x_1)$ with $f_1 = E$.

Assume $n > 1$. Let $G = (X, Y, A)$ be the bipartite multi-edges graph with $X = Y = \{0,1\}^{n-1}$ and A is the minimal set of edges such that for every x_n, y_n in $\{0,1\}$ and x, y in $\{0,1\}^{n-1}$ with $E(x_n, x) = (y_n, y)$, there is an edge $(x, y) \in A$ (that we label by $x_n y_n$). Obviously, every vertex of G has degree exactly two

since E is bijective. From the classical result of König (see [6],[7],[8]), the edges of G are colorable with two colors. Let us color them with elements of $\{0,1\}$. Now, define two mappings E^0, E^1 on $\{0,1\}^{n-1}$ and two mappings f_n, g_n from $\{0,1\}^n$ to $\{0,1\}$ as follow :

for each color $i \in \{0,1\}$ and every edge (x,y) of color i labelled $x_n y_n$, define :

$$\begin{aligned} E^i(x) &= y \\ f_n(x_n, x) &= i \\ g_n(i, y) &= y_n \end{aligned}$$

By construction, both mappings E^0, E^1 are bijective on $\{0,1\}^{n-1}$. Moreover, under induction hypothesis, each E^i is computed by a sequence of $2(n-1)-1$ assignments :

$$\begin{aligned} x_{n-1} &:= f^i_{n-1}(x_{n-1}, \ldots, x_1) \\ &\ldots\ldots \\ x_2 &:= f^i_2(x_{n-1}, \ldots, x_1) \\ x_1 &:= f^i_1(x_{n-1}, \ldots, x_1) \\ x_2 &:= g^i_2(x_{n-1}, \ldots, x_1) \\ &\ldots\ldots \\ x_{n-1} &:= g^i_{n-1}(x_{n-1}, \ldots, x_1) \end{aligned}$$

Define for every $i \in \{0,1\}$ and $x \in \{0,1\}^{n-1}$:

$$\begin{aligned} f_j(i,x) &= f^i_j(x) \text{ for } j = n-1, \ldots, 1 \\ g_j(i,x) &= g^i_j(x) \text{ for } j = 2, \ldots, n-1 \end{aligned}$$

in other words :

$$\begin{aligned} f_j(x_n, x) &= x_n.f^1_j(x) + (1+x_n).f^0_j(x) \text{ for } j = n-1, \ldots, 1 \\ g_j(x_n, x) &= x_n.g^1_j(x) + (1+x_n).g^0_j(x) \text{ for } j = 2, \ldots, n-1 \end{aligned}$$

By construction, the following program made of $2(n-1)-1+2 = 2n-1$ assignments transforms every vector $X = (x_n, \ldots, x_1)$ of $\{0,1\}^n$ into $E(X) = (y_n, \ldots, y_1)$:

x_n	$:= f_n(x_n, \ldots, x_1)$		$=$ a color value i
x_{n-1}	$:= f_{n-1}(x_n, \ldots, x_1)$	$= f^i_{n-1}(x_{n-1}, \ldots, x_1)$	Computes
x_{n-2}	$:= f_{n-2}(x_n, \ldots, x_1)$	$= f^i_{n-2}(x_{n-1}, \ldots, x_1)$	$E^i(x_{n-1}, \ldots, x_1)$
$\ldots$	$\ldots$	$\ldots$	by
x_2	$:= f_2(x_n, \ldots, x_1)$	$= f^i_2(x_{n-1}, \ldots, x_1)$	induction
x_1	$:= f_1(x_n, \ldots, x_1)$	$= f^i_1(x_{n-1}, \ldots, x_1)$	in
x_2	$:= g_2(x_n, \ldots, x_1)$	$= g^i_2(x_{n-1}, \ldots, x_1)$	$2(n-1)-1$
$\ldots$	$\ldots$	$\ldots$	steps
x_{n-2}	$:= g_{n-2}(x_n, \ldots, x_1)$	$= g^i_{n-2}(x_{n-1}, \ldots, x_1)$	
x_{n-1}	$:= g_{n-1}(x_n, \ldots, x_1)$	$= g^i_{n-1}(x_{n-1}, \ldots, x_1)$	Now $(x_{n-1}, \ldots, x_1) = (y_{n-1}, \ldots, y_1)$
x_n	$:= g_n(x_n, \ldots, x_1)$	$= g_n(i, y_{n-1}, \ldots, y_1)$	$= y_n$

and we are done. ■

Observe that the sequential decomposition of a bijective mapping E gives a practical tool to compute E, but also to compute E^{-1}. We just have to reverse the operations and read them from bottom to top.

3. General Case.

First, we will deal with a very special kind of mappings on $\{0,1\}^n$ that will be usefull for the general construction.

Definition. (step mapping). For every $M > 0$, a mapping f on $\{0,\ldots,M\}$ is a *step mapping* if for every $0 \leq x \leq y \leq M$:

$$0 \leq f(y) - f(x) \leq y - x$$

In particular one has $f(x) \leq f(x+1) \leq f(x) + 1$.

Definition. (simple). For every $n > 0$ and $x = (x_n,\ldots,x_3,x_2,x_1) \in \{0,1\}^n$, let $\phi(x) = 2^{n-1}.x_n + \ldots + 2^2.x_3 + 2.x_2 + x_1$. A mapping E on $\{0,1\}^n$ is *simple* if the mapping E' on $[0,2^n-1]$ such that

$$E(x) = y \Longleftrightarrow E'(\phi(x)) = \phi(y)$$

is a step mapping.

Proposition (direct). *For every $n > 0$, every simple mapping E on $\{0,1\}^n$ is computed by a program of the form :*

$$\begin{aligned} x_1 &:= p_1(x_n,\ldots,x_1) \\ x_2 &:= p_2(x_n,\ldots,x_1) \\ &\ldots\ldots \\ x_n &:= p_n(x_n,\ldots,x_1) \end{aligned}$$

Proof. Since the number of assignments is minimal, necessarily each function p_i must give its correct final value to each component x_i. That is to say, for $E(x_n,\ldots,x_1) = (y_n,\ldots,y_1)$ and for each $i = 1,2,\ldots,n$:

$$p_i(x_n,\ldots,x_i,y_{i-1},\ldots,y_1) = y_i$$

It remains to prove that this unique possible method is correct. Assume the converse. Hence, two different vectors x, x' become, after some step i, the same vector z whereas their final images $y = E(x)$ and $y' = E(x')$ are different.

Hence, $x_n = x'_n = z_n, \ldots, x_{i+1} = x'_{i+1} = z_{i+1}$ and for $a = \phi(x)$ and $b = \phi(x')$ (assume that $b > a$) :

$$b - a < 2^i$$

In the other hand, since they become the same vector, one has $y_i = y'_i = z_i, \ldots, y_1 = y'_1 = z_1$. For $c = \phi(y)$ and $d = \phi(y')$, one has $d - c \geq 2^i$. A contradiction with the hypothesis that E' is a step mapping which implies : $d - c \leq b - a$ ■

Now we can deal with the general case, that is to say, mappings E on $\{0,1\}^n$ for which some different vectors x, x' may have same images. First, we group the vectors with same images via a bijective mapping G. The aim is to give intermediary consecutive images in lexicographical order to vectors with same final images. The second step consists in the identification of vectors via a simple mapping P. The third step consists in the attribution of the correct final image value to each vector via another bijective mapping F.

Definition. (decomposition). For $n > 0$, let E be a mapping on $\{0,1\}^n$. The *decomposition* of E is a triple of mappings (F, P, G) on $\{0,1\}^n$ defined by the following program :

`0.` `Begin with` $G(x) = P(x) = F(x) =?$ `for every` $x \in \{0,1\}^n$ `and with` $K = k = 0$.

`1.` `While there is some` $G(x) =?$ `do`

`1.1.` `For every` x' `with` $E(x') = E(x)$ `do`

`1.1.1.` $G(x') = \phi^{-1}(k)$

`1.1.2.` $P(G(x')) := \phi^{-1}(K)$

`1.1.3.` $F(P(G(x'))) := E(x)$

`1.1.4.` $k := k + 1$

`1.2.` $K := K + 1$

`2.` `Complete the definition of` F `such that` F `is bijective.`

4. An example.

For $n = 3$:

x	$E(x)$	$G(x)$	k	$P(G(x))$	K	$F(P(G(x)))$
000	111	000	0	000	0	111
001	000	101	5	001	1	000
010	111	001	1	000	0	111
011	000	110	6	001	1	000
100	111	010	2	000	0	111
101	000	111	7	001	1	000
110	111	011	3	000	0	111
111	111	100	4	000	0	111

Completion of F :

x	$F(x)$
000	111
001	000
010	010
011	011
100	100
101	101
110	110
111	001

By construction F and G are bijective and P is simple.

Hence, by composition, one obtains a program that computes $X := E(X)$ for the given mapping E in $(2n-1)+(n)+(2n-1) = 5n-2$ assignments.

One can perform two successive operations on a same element in a single one.

By this way, the $5n-2$ assignments are reduced to $5n-4$ assignments.

Improvements :

Computation with No Memory and Rearrangeable Multicast Networks
Discrete Mathematics and Theoretical Computer Science
DMTCS vol.16:1, 2014, 121142