Nouveaux Ordinateurs Résilients.

Nous ne perdons pas le NOR et gardons le CAP $\cap$ vers des ordinateurs nouveaux.

Nous allons essayer d'implémenter la fonction NOR dans des ordinateurs "affaiblis" de sorte que la puissance de l'algèbre linéaire puisse permettre d'effectuer des calculs optimaux. Nous poursuivons ces idées concernant la fonction logique :

$$NOR(x,y) = \begin{cases} \mathbf{1} \text{ si } x{=}0,\ y{=}0 \\ \mathbf{0} \text{ si } x{=}0,\ y{=}1 \\ \mathbf{0} \text{ si } x{=}1,\ y{=}0 \\ \mathbf{0} \text{ si } x{=}1,\ y{=}1 \end{cases}$$

Le but est d'implémenter cette fonction par une séquence d'opérations affines sur des registres faibles de sorte que leurs "erreurs" d'arrondis deviennent des qualités pour atteindre nos objectifs. Nous introduisons un formalisme qui nous servira plus tard mais pas vraiment ici. Pour $0 \leq d \leq n$, un R_d^n est un registre de n bits où d bits représentent la partie fractionnaire d'un nombre rationnel. Ainsi un R_d^n ayant comme contenu $b_{n-1} \dots b_0$ représente le nombre $b_{n-1}2^{n-1-d} + \dots + b_0 2^{-d}$. Comme exemple 1011 représente "onze" dans un R_0^4 et "deux virgule soixante quinze" dans un R_2^4.

Nous allons considérer ici des R_0^2. Ces registres sont très faibles mais en fait ils seront très puissants pour ce que l'on souhaite faire. Soit A un R_0^2. Si $A{=}3$, alors $A/2{:=}1$ et $A{+}1{:=}0$. Ces deux calculs sont faux mais ces erreurs rendront corrects nos futurs calculs pour la fonction NOR. Ainsi, une faiblesse devient une force. Il était impensable que la fonction NOR puisse se calculer avec des fonctions affines mais la faiblesse de ces R_0^2 permettra d'obtenir ce résultat. Les erreurs de calculs corrigent la non affinité.

Soit $a(x,y){:=}x{+}y{+}3$. Calculons avec des R_0^2 la fonction $a(x,y)$ sur les valeurs $0, 1$:

$$a(x,y) = \begin{cases} 3 = \mathbf{11} \text{ si } x{=}00,\ y{=}00 \\ 4 = \mathbf{00} \text{ si } x{=}00,\ y{=}01 \\ 4 = \mathbf{00} \text{ si } x{=}01,\ y{=}00 \\ 5 = \mathbf{01} \text{ si } x{=}01,\ y{=}01 \end{cases}$$

Maintenant, calculons "comme un humain" $h(x,y){:=}(x{+}y{+}3)/2$ toujours dans des R_0^2 :

$$h(x,y) = \begin{cases} 1.5 = \mathbf{01} \text{ si } x{=}00,\ y{=}00 \\ 2.0 = \mathbf{10} \text{ si } x{=}00,\ y{=}01 \\ 2.0 = \mathbf{10} \text{ si } x{=}01,\ y{=}00 \\ 2.5 = \mathbf{10} \text{ si } x{=}01,\ y{=}01 \end{cases}$$

Mais nous avons fait de l'anthropocentrisme. Nous avons calculé puis mis le résultat dans le R_0^2. Une machine ne fera jamais cela. Elle est plus bête mais aussi plus méthodique qu'un humain. Elle effectue les calculs en suivant l'ordre des parenthésages. D'abord elle effectue les additions, puis elle effectue la division. Et c'est encore cette faiblesse qui sera une force puisque l'on obtient à partir de $a(x,y)$ et après division par 2 une fonction $n(x,y)$:

$$n(x,y) = \begin{cases} 1 = \mathbf{01} \text{ si } x{=}00, y{=}00 \\ 0 = \mathbf{00} \text{ si } x{=}00, y{=}01 \\ 0 = \mathbf{00} \text{ si } x{=}01, y{=}00 \\ 0 = \mathbf{00} \text{ si } x{=}01, y{=}01 \end{cases}$$

Et nous avons obtenu notre fonction logique complète NOR, calculée avec des opérations affines.

Remarquez que le $NAND$ s'obtient également avec une fonction affine sur des R_0^2 : $(x+y+2)/2$.

Claim. *Toute fonction logique à n variables est de la forme affine $(a_0 + a_1x_1 + \dots + a_nx_n)$ calculée sur un R_0^m où les a_i sont des nombres rationnels.*

(S.I.G.L.E.) Special Investigation Group for Life on Earth. www.sigle.space