Realizzabilità e altre cose interessanti

Alberto Fiori

Abstract. As I understand it the speciality of a realizzability interpretation is in being looser, for example in this theory the term $natrec(x, 0, 0)$, where x is a free variable, belongs to every type despite not even being well formed in a type theory context.

1. Copycatting: Let's get vaguely serious

1.1. Starting definitions. We are going to give a different interpretation of the same language used in the Minimalist Foundation. Una delle differenza sarà che definiremo un'untyped relazione di riducibilità invece del giudizio di uguanza tipata; then using the distinction in **canonical** and **non-canonical** expression we will define the set of **normal** expressions (le espressioni normali in mtt sono normali anche qui) Let's give a simple grammar for our language (in this interpretation we don't actually diffenrentiate between types and elements): let $\mathcal{V}$ be an infinite set of variables (generally denoted with x, y, z, w and with [1]

Definition 1.1. The set of expression $\mathcal{E}$ will be inductively genererated from the set of variables $\mathcal{V}$ as closed under the following construct:

$$\lambda x.e \tag{1}$$

$$apply(e_1, e_2) \tag{2}$$

$$0 \tag{3}$$

$$succ(e) \tag{4}$$

$$natrec(e_1, e_2, e_3) \tag{5}$$

$$< e_1,\ e_2 > \tag{6}$$

$$El_\Sigma(e_1, e_2) \tag{7}$$

$$N \tag{8}$$

$$\Sigma(e_1, e_2) \tag{9}$$

$$\Pi(e_1, e_2) \tag{10}$$

where x is a variable and each e_i is a previously constructed expression.

Definition 1.2. An expression is said to be **canonical** if it is in the form $\lambda x.e$, 0, $succ(e)$, $< e_1, e_2 >$

Definition 1.3. the **contraction** relation ($\triangleright$) is defined by:

$$natrec(0, b, c) \triangleright \qquad b \tag{11}$$

$$apply(\lambda x.e, d) \triangleright \qquad b\,\{a/c\} \tag{12}$$

$$el \tag{13}$$

[1]pensando di nuovo a come trattare il concetto delle *astrazioni* in *proglöf* credo che per ora la cosa migliore sia dimenticarsene e non usarle

Definition 1.4. Here we give the definition of ψ and Ψ as inference rules:

$$\frac{}{\Psi(N)} \tag{14}$$

$$\frac{\Psi(A) \qquad \Psi(B(a))\,[\psi(A,h,a)]}{\Sigma(A,B)} \tag{15}$$

$$\frac{\Psi(A) \qquad \Psi(B(a))\,[\psi(A,h,a)]}{\Pi(A,B)} \tag{16}$$

$$\frac{\mathcal{F}(A)}{\Psi(A)} \quad \frac{A \to B \qquad \Psi(B)}{\Psi(A)} \tag{17}$$

$$\frac{a \to b \qquad \mathcal{N}(b)}{\psi(N,a)} \tag{18}$$

$$\frac{c_{1b}\colon \Psi(\Pi(A,B)) \qquad \psi(B(a), apply(b,a))[\psi(A,a)]}{\psi(\Pi(A,B), c_?, b)} \tag{19}$$

$$\frac{c_{1c}\colon \Psi(A) \qquad a \to b \qquad \mathcal{F}(b)}{\psi(A,a)} \tag{20}$$

$$\frac{c_{1d}(A,B,h)\colon \Psi(A) \qquad \psi(B,a)}{\psi(A,a)} \tag{21}$$

$$\frac{}{\Phi(U)} \quad \frac{\Psi(A)}{\phi(U,A)} \tag{22}$$

The normalization theorem will be proved in two steps. In the first step we will prove basic properties of this *Realizzability Interpretation* (such as being normalizing and having other computational properties). In the second step we will give a proof of compatibility between the Minimalist Foundation and the

Lemma 1.1 (qwe).

CHAPTER 1

Expression Language

The following is based on the ideas that I first read in the book "programming in Martin Löf type thery" [1]. Basically every expression we write in a mathematical expression has an arity (0,$\alpha \to \beta$, $\alpha_1 \otimes \alpha_2 \ldots \otimes \alpha_n$) which work similarly to how types in simply typed lambda calculus work. This simply is a way to ensure that expression are well-formed (even if it doesn't ensure that they are reasonable); we chose to add the $\otimes \ldots \otimes$ constructor to freely chose when to *curry* and *uncurry* function application; we could have added more, but there seem to be no advantage in doing so.

The rule for arity are the expected ones:

1. Everithhung has an Arity

123123123123

2. The previous section was a bad idea

As already pointed out in the footnote 1 on page 4 it was not a good idea to start from the wellformedness of formulas as to prove the normalization theorem we need just the concepts of **canonical** and **non-canonical** and the **tree-like** structure of expressions to define the reducibility relation ricursively.

In the whole article, I think, nothing else was used.

[1]Non sono più convinto di questa cosa; nel libro si parla escusivamente di $MLTT$ dove c'è un enorme rigidità nelle costruzioni ammesse, al contrario quando siamo in realizzabilità abbiamo decisamente una maggior libertà: potremmo per esempio scrivere l'espressione apply$(0, 0)$ semplicemente questa non farebbe parte di alcun tipo. rimarrebbe ora da decidere come trattare il concetto di arietà funzionale, la Coquand definisce il costruttore di funzione come $\lambda x.e$ dando nomi espliciti alle variabili e richiamandosi all'operatore di sostituzione $B\{a/x\}$ per rappresentare i "conti" seguendo questa struttura, la consueta sintassi di funzione diventa obsoleta, il lambda calcolo funzionale (i.e. $(x).f$ o $\langle x \rangle .f$) diventa di fatto non strettamente necessario negli usi che servono nell'articolo.
Questo mi porta a propormi di non accanirmi sulle a rietà di non preoccuparsi troppo del $\lambda x.e$ dei Π-tipi (a cui n on sono troppo abituato: $\lambda x.e = \lambda((x).e)$)