# 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

Definition 1.1. The set of expression $\mathcal{E}$ will be inductively generated from the set of variables $\mathcal{V}$ as

pensando di nuovo a come trattare il concetto delle *astrazioni* in *proglöf* credo che per ora la cosa migliore sia dimenticarsene

closed under the following construct:

(1) $\lambda x.e$

(2) $apply(e_1, e_2)$

(3) $0$

(4) $succ(e)$

(5) $natrec(e_1, e_2, e_3)$

(6) $< \cdot, \cdot > e_1, e_2$

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

(8) $N$

(9) $\Sigma(e_1, e_2)$

(10) $\Pi(e_1, e_2)$

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:

(11) $natrec(0, b, c) \triangleright \quad b$

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

(13)

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

[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 $\mathrm{apply}(0,0)$ 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: $\lambda x.e = \lambda((x).e)$)

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 **??** on page **??** 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.