# A proposal for universe management in Coq

**Vladimir Voevodsky**

Started January 11, 2012

# Contents



## 1 Introduction

We start by a description of the system of expressions underlying the purposed type system. We will provide this description on the semi-syntactic level with named variables.

Semi-syntactic means here that we are going to define expressions as planar trees with vertices marked either by a name of a variable or by a u-level expression (see below) or by a sequence of the form $(S; \vec{a}_1; \dots; \vec{a}_n)$ where $S$ is a special symbol (from the given set of allowed special symbols), $n \geq 0$ and for each $i = 1, \dots, n$, $\vec{a}_i$ is (possibly empty) sequence of names of variables. The occurrences of these variables in the nodes below such a node are considered to be bound by the node. All expressions are considered up to $\alpha$-equivalence i.e. the renaming of bound variables and since the problems related to the possible conflict of variable names are the usual ones we ignore them in this exposition.

This level of formalization is an intermediate one between the syntactic (linear) presentation of expressions on the one hand and nameless presentation (using some version of de Brujin indexes) on the other. The choice of a particular syntactic representation which is necessary at the level of the user interface and of a particular nameless representation which is necessary for the internal manipulation of expressions by the system can vary and are left for the next stage of concretization.

On notations: we will write $g[a]$ to indicate that $g$ is an expression which may contain free variable $a$ and $g[E/a]$ for the expression obtained by the substitution of the expression $E$ instead of $a$ in $g$.

We will write $\vec{a}$ for a sequence $a_1, \dots, n$ and let $dim(\vec{a})$ denote the number of items in the sequence.

We assume two alphabets for variables one for u-level variables and one for usual variables. Unless otherwise indicated variables named $x, y, z$ (with or without diacritics) are usual variables and the ones named $p, q, r$ are u-level variables (see below). The letters $i, j, k, l, m, n$ are reserved for numerals (e.g. in indexes).

We write $[L](B_1, \dots, B_n)$ for a labelled (planar i.e. with ordered branches) tree with the root node labelled by $L$ and branches $B_1, \dots, B_n$. We also abbreviate $[L]([L'](B_1, \dots, B_n))$

as $[L][L'](B_1, \ldots, B_n)$.

## 2 u-level expressions and admissible domains

**Definition 2.1** *[**d21a**] A u-level expression is either a numeral or a u-level variable or an expression of the form $M + n$ where $M$ is a u-level expression and $n$ is a numeral.*

Two u-level expressions with variables from a finite set $Fu$ are considered to be definitionally equal if they represent the same functions on $\mathbf{N}^{Fu}$. Up to definitional equality any u-level expression if either a numeral or the sum of a u-level variable with a numeral.

**Definition 2.2** *[**d21b**] Let $Fu$ be a finite set. For $p \in Fu$ let $x_p$ be the corresponding coordinate function on $\mathbf{N}^{Fu}$. A subset $A$ of $\mathbf{N}^{Fu}$ is called admissible if it belongs to the smallest class of subsets satisfying the following conditions:*

1. *subset defined by an equation of the form $x_p = n$ where $p \in Fu$, $n \in \mathbf{N}$ is admissible,*
2. *subset defined by an equation of the form $x_p = x_q + n$ where $p, q \in Fu$ and $n \in \mathbf{N}$ is admissible,*
3. *subset defined by an inequality of the form $x_p \geq y_q + n$ or an inequality of the form $x_p \leq y_q + n$ where $p, q \in Fu$ and $n \in \mathbf{N}$ is admissible,*
4. *if subsets $A_1, A_2$ are admissible then $A_1 \cap A_2$ is admissible.*

It will be important for us that there exists an efficient decision procedure which determines whether or not the admissible subset defined by a given system of equations and inequalities of Definition 2.2 is empty.

## 3 System CPC

We first describe the sub-system $CPC$ (calculus of polymorphic constructions) of our type system which has elements related to universes and to dependent products and later add the elements related to other constructions.

### Terms of CPC

**Definition 3.1** *[**d22**] The following labels are permitted in the labelled trees of $CPC$ : names of variables, u-level expressions, $el$, $(\prod; x)$, $\mathcal{U}$, $J$, $ev$, $(\lambda; x)$, $forall$.*

**Definition 3.2** *[**d25**] We distinguish three classes of labelled trees:*

1. *u-level expressions,*

2. *o-trees which are trees whose root node is labelled by el or* $(\prod; x)$,

3. *s-trees which are trees whose root node is labelled by the name of a variable,* $\mathcal{U}$, $J$, $forall$, $ev$ *or* $(\lambda; x)$.

**Definition 3.3** *[***d20***] A CPC-expression is a labelled planar tree with labels from the set described in Definition 3.1 which satisfy the following conditions:*

1. *any node of the form* $[M]$ *where* $M$ *is a u-level expression has valency* 0 *(i.e. is a leaf),*

2. *any node of the form* $[el]$ *has valency* 1 *and its only branch is an s-tree,*

3. *any node of the form* $[\prod; x]$ *has valency* 2 *and both of its branches are o-trees,*

4. *any node of the form* $[x]$ *where* $x$ *the name of a (usual) variable has valency* 0 *(i.e. is a leaf),*

5. *any node of the form* $[\mathcal{U}]$ *has valency* 1 *and its only branch is a node labelled by a u-level expression,*

6. *any node of the form* $[J]$ *has valency* 2 *and both of its branches are nodes labelled by u-level expressions,*

7. *any node of the form* $[forall]$ *has valency 1 and its only branch is a node labelled by a u-level expression,*

8. *any node of the form* $[ev]$ *has valency* 2 *and both of its branches are s-trees,*

9. *any node of the form* $[\lambda; x]$ *has valency 2, its first branch is an o-tree and its second branch is an s-tree,*

Intuitively, u-level expressions correspond to universe levels, o-expressions correspond to types and s-expressions correspond to terms.

**Lemma 3.4** *[***l20***] One has:*

1. *Any branch of a CPC-expression is a CPC-expression,*

2. *modulo possible necessity to change the names of bound variables to avoid name conflicts, the labelled tree obtained by a replacement of any node of the form* $[x]$ *where* $x$ *is the name of a (usual) variable by an s-expression is again a CPC-expression.*

**Proof**: Straightforward.

We let $CPC$ denote the set of CPC-expressions and $UE$, $OE$ and $SE$ denote the subsets of u-level expressions, o-expressions and s-expressions in CPC.

According to the general rule stated above, variables in the expression which occur in its nodes labelled by $(\prod, \_)$ and $(\lambda, \_)$ are called bound. The other ones are called free. We will write $CPC(Fv, Fu)$, $OE(Fv, Fu)$ and $SE(Fv, Fu)$ for the sets of expressions, o-expressions and s-expressions respectively with free variables from the set $Fv$ and u-level variables from the set $Fu$.

For $E \in CPC(Fv, Fu)$ and an element $\vec{n} \in \mathbf{N}^{Fu}$ we write $E[\vec{n}]$ for the CPC-expression obtained by evaluating all of the u-level expressions in $E$ on $\vec{n}$.

For a linearly ordered set $Fv$ we will write $Fv \amalg \{x\}$ for the disjoint union of $Fv$ with a one element set $\{x\}$ ordered such that $a < x$ for all $a \in Fv$.

**Definition 3.5** *[**d23**] Let $E$ be an expression with free variables in a linearly ordered set $Fv$ and u-variables in a set $Fu$. Let $\phi : Fv \to OE$ be a function assigning an o-expression to each free variable of $E$. Let $A$ be an admissible domain in $\mathbf{N}^{Fu}$. We define the notions of an expression strictly well formed relative to $\phi$ and $A$ and function $\tau = \tau(\phi)$ from strictly well formed s-expressions to $OE(Fv, Fu)$ recursively as follows:*

1. *A u-level expression $E$ is strictly well formed.*
2. *An expression of the form $[el](E')$ is strictly well formed iff $E'$ is a strictly well-formed s-expression relative to $\phi$, $A$ and $\tau(E') = [el][\mathcal{U}][N]$ for an u-level expression $N$.*
3. *An expression of the form $[\prod; x](E_1, E_2)$ is strictly well formed iff:*
   (a) *$E_1$ does not contain $x$ and is a strictly well formed o-expression relative to $\phi$, $A$,*
   (b) *$E_2$ is a strictly well formed o-expression relative to the function $\phi'$ on $Fv \amalg \{x\}$ which equals $\phi$ on $Fv$ and $E_1$ on $x$ and $A$.*
4. *For $x \in Fv$, $[x]$ is strictly well formed iff:*
   (a) *$\phi(x) \in OE(\{x' \in Fv \mid x' < x\})$,*
   (b) *$\phi(x)$ is strictly well formed relative to the restriction of $\phi$ to $\{x' \in Fv \mid x' < x\}$ and $A$,*

   *under these conditions $\tau([x]) = \phi(x)$.*
5. *An expression of the form $[\mathcal{U}](M)$ is strictly well formed and $\tau(\mathcal{U}) = [el][\mathcal{U}][M + 1]$.*
6. *An expression of the form $[J](M_1, M_2)$ is strictly well formed if for all $\vec{n} \in A$ one has*

$$M_1(\vec{n}) \leq M_2(\vec{n})$$

   *under this condition*

$$\tau([J](M_1, M_2)) = [\prod; x]([el][\mathcal{U}][M_1], [el][\mathcal{U}][M_2])$$

7. *An expression of the form $[forall][M]$ is strictly well formed and*

$$\tau([forall][M]) = [prod; x]([el][\mathcal{U}][M], [\prod; F]([\prod; y]([el][x], [el][\mathcal{U}][M]), [el][\mathcal{U}][M]))$$

8. *An expression of the form* $[ev](E_1, E_2)$ *is strictly well formed iff:*

   (a) $E_1$ *is a strictly well formed s-expression relative to* $\phi$, $A$ *and* $\tau(E_1) = [\prod; x](E_{1,1}, E_{1,2})$,

   (b) $E_2$ *is a strictly well formed s-expression relative to* $\phi$, $A$ *and for all* $\vec{n} \in A$ *one has* $\tau(E_2)[\vec{n}] = E_{1,1}[\vec{n}]$

   *under these conditions*

$$\tau([ev](E_1, E_2)) = E_{1,2}[E_2/x]$$

9. *An expression of the form* $[\lambda; x](E_1, E_2)$ *is strictly well formed iff:*

   (a) $E_1$ *does not contain* $x$ *and is a strictly well formed o-expression relative to* $\phi$, $A$,

   (b) $E_2$ *is a strictly well formed s-expression relative to the function* $\phi'$ *on* $Fv \amalg \{x\}$ *which equals* $\phi$ *on* $Fv$ *and* $E_1$ *on* $x$ *and* $A$,

   *under these conditions*

$$\tau([\lambda; x](E_1, E_2)) = [\coprod; x](E_1, \tau(\phi', E_2))$$

**Remark 3.6** *[***rem1***]*It is easy to see that for given $E$ and $\phi$ there is a largest admissible subset $A$ such that $E$ is strictly well formed relative to $\phi$ and $A$. We denote this subset by $udom(E, \phi)$.

We let $SWFCPC(Fu, Fv, \phi, A)$, $SWFOE(Fu, Fv, \phi, A)$ and $SWFSE(Fu, Fv, \phi, A)$ denote the sets of expressions, o-expressions and s-expressions from $CPC(Fu, Fv)$ which are strictly well formed relative to $\phi$ and $A$.

**Lemma 3.7** *[***l21***] Let* $E \in SWFCPC(Fu, Fv, \phi, A)$ *be a strictly well formed expression,* $x \in Fv$ *and* $E' \in CPC(Fu, Fv - \{x\})$ *an s-expression which does not contain* $x$*, is strictly well formed relative to* $\phi$ *and* $A$ *and such that* $\tau(E') = \phi(x)$*. Then* $E[E'/x]$ *is strictly well formed relative to* $\phi$ *and* $A$*.*

**Proof**: ???

**Theorem 3.8** *[***th1***] For any* $E \in SWFCPC(Fu, Fv, \phi, A)$ *one has* $\tau(E) \in SWFOE(Fu, Fv, \phi, A)$.

**Proof**: ???

**Reduction rules for CPC** A strictly well formed relative to $\phi$, $A$ CPC-expression with u-level variables from $Fu$ is called a root-node redux relative to an admissible subset $A$ if it has one of the following forms:

1. $[el][ev]([J]([M_1], [M_2]), X)$ (reduces to $[el](X)$, possibly increasing $udom$),

2. $[el][ev]([ev]([forall][M], X), F)$ (reduces to $[\prod; x]([el](X), [el][ev](F, X))$, no change in *udom*),

3. $[ev]([\lambda; x](X, f), a)$ (reduces to $f[a/[x]]$ with a possibly increasing *udom*),

4. $[ev]([J]([[M_1], [M_2]), X)$ (reduces to [X] possibly reducing *udom* by the condition $M_1(\vec{n}) = M_2(\vec{n})$),

5. $[ev]([J]([M_3], [M_4]), [ev]([J]([M_1], [M_2]), X))$ (reduces to $[ev]([J]([M_1], [M_4]), X)$ possibly increasing *udom*),

6. $[ev]([J]([M_1], [M_2]), [ev]([ev]([forall][M_3], X), F)$ (reduces to $[ev]([ev]([forall][M_2], [ev]([J]([M_3], [M_2]), X)), [\lambda; a]([el][X], [ev]([J]([M_3], [M_2]), [ev](F, X))))$ with changes in *udom*),

7. $[\lambda; x](X, [ev](f, [x]))$ (reduces to $f$, no change in *udom*).

**Adding dependent sums - system CPC1**

**Definition 3.9** *[**d32**] The following labels are permitted in the labelled trees of $CPC1$ - the labels permitted in CPC, $(\sum; x)$, pair, $pr1$, $pr2$, $[total]$.*

The notions of u-level expressions, o- and s- trees in CPC1 are defined as follows:

**Definition 3.10** *[**d35**]*

1. *trees with the root node caring a CPC-label is an u-level expression, s-tree or an o-tree according to the rules of Definition 3.2,*

2. *trees with the root of the form $[\sum; x]$ are o-trees,*

3. *trees with the root node of the form $[pair]$, $[pr1]$, $[pr2]$ and $[total]$ are s-trees.*

**Definition 3.11** *[**d30**] A CPC1-expression is a CPC1-labelled planar tree such that:*

1. *any node caring one of the CPC-labels satisfies the conditions of Definition 3.3,*

2. *any node of the form $[\sum; x]$ has valency 2 and both its branches are o-trees,*

3. *any node of the form $[pair]$ has valency 2,*

4. *any node of the form $[pr1]$ has valency 2,*

5. *any node of the form $[pr2]$ has valency 2,*

6. *any node of the form $[total]$ has valency 1 and its only branch is a node labelled by a u-level expression.*

The obvious analog of Lemma 3.4 holds for CPC1-expressions. We extend to CPC1 the basic notations introduced in the context of CPC.

**Definition 3.12** *[**d33**] Under the obvious CPC1-analog of assumptions and notations of Definition 3.5 we extend the notion of being strictly well formed, the function $\tau$ and the function udom to CPC1-expressions recursively as follows:*

1. *an expression whose root node carries a CPC-label is strictly well formed if it satisfies the obvious analog of the corresponding condition of Definition 3.5.*
2. *an expression of the form $[\sum; x](E_1, E_2)$ is strictly well formed iff* [1]
   (a) *$E_1$ does not contain $x$ and is a strictly well formed o-expression relative to $\phi$,*
   (b) *$E_2$ is a strictly well formed o-expression relative to the function $\phi'$ on $Fv \amalg \{x\}$ which equals $\phi$ on $Fv$ and $E_1$ on $x$,*

   *under these conditions $udom([\prod; x](E_1, E_2)) = udom(E_1) \cap (udom(phi', E_2))$,*
3. *an expression of the form $[pair](E_1, E_2)$ is strictly well formed iff:*

[1] It is exactly the same rule as for $[\prod; x]$