Notes on Universes in Type Theory[*]

Zhaohui Luo[†]

October 2012

1 Introduction

Martin-Löf [ML84] introduced two styles of universes in type theory: the Tarski-style and the Russell-style. The Tarski-style universes are semantically more fundamental but the Russell-style universes are easier to use in practice. It would be interesting to ask

- whether these two styles of universes are equivalent, and
- whether we can employ and implement universes in a Russell-style that faithfully represent the Tarski-style universes.

The answer to the first question is negative, as long as we consider the *current formulation*[1] of the intensional type theories (and assume that a type theory with Tarski-style universes have nice meta-theoretic properties). In fact, as explained in this note, the Russell-style universes seem incompatible with the general form of elimination rules of inductive types – 'incompatible' in the sense that the canonicity property and the subject reduction property fail to hold in the resulting type theory.[2]

[*]Notes for a talk at Institute for Advanced Study in Princeton.

[†]zhaohui.luo@hotmail.co.uk, Royal Holloway, Univ of London.

[1]The emphasis may be important. See a discussion at the end of §3 on additional definitional equality rules.

[2]A key rule for the Russell-style universes is a special case of *subsumptive subtyping*. The problem is indeed a special case for subsumptive subtyping when it is employed in intensional type theories with inductive types. It is pointed out in §4 of [LSX12] that there are two different views on typing, one for type assignment systems such as those employed in functional programming languages and the other for type theories with canonical objects such as those implemented in proof assistants, and these two views on typing correspond to two different views on subtyping: subsumptive subtyping for type assignment systems and coercive subtyping for TTs with canonical objects. See the paper for more details.

This negative answer to the first question seems to imply that the answer to the second question must be negative, too. However, it is argued that the essence of the Russell-style universes can be obtained by means of the Tarski-style universes *together with coercive subtyping* in taking the explicit lifting operators between universes as coercions (plus some notational conventions). In this way, one may obtain both the theoretical adequacy and the easy-to-use benefit.

2 Type Universes and Cumulativity

A universe is a type of types. One may consider a sequence of universes indexed by natural numbers[3]: intuitively,

$$U_0 \in U_1 \in U_2 \in ...$$

$$U_0 \subseteq U_1 \subseteq U_2 \subseteq ...$$

Tarski-style Universes. The Tarski-style universes are introduced by

$$\frac{}{U_i\ type} \qquad \frac{a : U_i}{T_i(a)\ type} \qquad \frac{a : U_i}{t_{i+1}(a) : U_{i+1}}$$

$$\frac{}{u_i : U_{i+1}} \qquad \frac{}{T_{i+1}(u_i) = U_i} \qquad \frac{a : U_i}{T_{i+1}(t_{i+1}(a)) = T_i(a)}$$

together with the rules for inductive types, examples of which are:

$$\frac{}{nat : U_0} \qquad \frac{}{T_0(nat) = Nat}$$

$$\frac{a : U_i}{list(a) : U_i} \qquad \frac{a : U_i}{T_i(list(a)) = List(T_i(a))}$$

Russell-style universes. The Russell-style universes are introduced by

$$\frac{}{U_i\ type} \qquad \frac{A : U_i}{A\ type} \qquad \frac{}{U_i : U_{i+1}} \qquad \frac{A : U_i}{A : U_{i+1}}$$

together with rules for inductive types, examples of which are

$$\frac{}{Nat : U_0} \qquad \frac{A : U_i}{List(A) : U_i}$$

[3]I mainly consider universes indexed by natural numbers in this note. Universes indexed by level expressions are briefly discussed in a remark at the end of §4. For further universe operators and super universes, see [Pal98].

Remark

- Note that the fourth rule for Russell-style universes is equivalent to $U_i \leq U_{i+1}$ plus the following subsumption rule:

$$\frac{a : A \quad A \leq B}{a : B}$$

- The Tarski-style rules can be formulated in a logical framework:

$$\begin{array}{rl} U_i : \textbf{Type} & u_i : U_{i+1} \\ T_i : (U_i)\textbf{Type} & T_{i+1}(u_i) = U_i : \textbf{Type} \\ t_{i+1} : (U_i)U_{i+1} & T_{i+1}(t_{i+1}(a)) = T_i(a) : \textbf{Type} \end{array}$$

  but the Russell-style rules cannot.

Cumulativity for Structural Subtyping The basic subtyping relations such as those concerning universes should be propagated through the type constructors. For example, because $U_0 \leq U_1$, we should have $U_0 \times U_0 \leq U_1 \times U_1$. As another example, for $List(A)$, we naturally have:

$$\frac{A \leq B}{List(A) \leq List(B)}$$

For coercive subtyping, this amounts to the following rule:

$$\frac{A \leq_c B}{List(A) \leq_{map(A,B,c)} List(B)}$$

where $map(A, B, c)$ is the usual map-operation on lists defined by induction as follows:

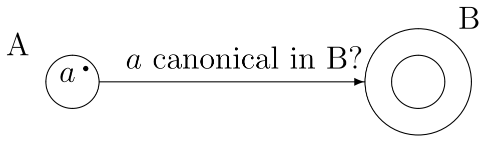
$$\begin{array}{rcll} map(A, B, nil(A)) & = & nil(B) & (1) \\ map(A, B, cons(A, a, l) & = & cons(B, c(a), map(A, B, l)) & (2) \end{array}$$

3 Problems with the Russell-style Universes

Are the Russell-style universes (and, in general, subsumption) compatible with the general form of elimination rules for inductive types? The answer is negative as the following two examples show.

Canonicity. The property of canonicity states that any closed object of an inductive type is definitionally equal to a canonical object of that type. It is a basis to justify the induction principles (elimination rules) for inductive types.

Considering the Russell-style universes, we may ask: if $A \leq B$ and a is a canonical object of A, is a a canonical object of B?



Here is an example to show that canonicity fails.

Example 3.1 *Let A and B be closed types such that $A \leq B$ but $A \neq B$. Then $nil(A) : List(A)$ and, by subsumption, $nil(A) : List(B)$. But $nil(A)$ is not definitionally equal to any canonical object of $List(B)$, $nil(B)$ or $cons(B, b, l)$. Canonicity fails to hold.*

This has other unpleasant consequences. For instance, we can prove the following proposition by means of the elimination rule for $List(B)$:

$$Id(List(B), nil(A), nil(B)) \vee \exists b{:}B \exists l{:}List(B).\ Id(List(B), nil(A), cons(B, b, l)).$$

but neither of the disjuncts is the case definitionally. Therefore, the property of equality reflection does not hold: there is a mismatch between the definitional and propositional equalities (even for closed terms).

Remark An intuitive understanding of this situation is that subsumptive subtyping has introduced new objects into a type ($nil(A)$ is introduced in $List(B)$ in the above example), which are not covered by the elimination rules. One would ask: are these new objects canonical objects? (See the above picture.) If they are, the traditional elimination rule does not cover them and, in order to cover them, one might even consider an elimination rule with 'bounded quantification' (take the type of lists as an example):

$$\begin{aligned}
\mathcal{E}'_{List} \quad : \quad & (A : \mathbf{Type})(A_0 \leq A : \mathbf{Type}) \\
& (C : (List(A_0))\mathbf{Type}) \\
& (c : C(nil(A_0))) \\
& (f : (a : A_0)(l : List(A_0))C(cons(A_0, a, l))) \\
& (z : List(A_0))C(z)
\end{aligned}$$

But bounded quantification is too much and problematic. (I omit discussions here.)

Subject Reduction. The property of subject reduction, for a reduction relation $\rhd$, is that, if $a : A$ and $a \rhd b$, then $b : A$. It is an important property, both theoretically in meta-theory and practically in implementations. Unfortunately, subject reduction does not hold for non-constant (parameterised) inductive type constructors, if Russell-style universes are employed. Here, the example is about the cartesian product types and can be found in §4.3 of [Luo99].

Example 3.2 *Consider the following context:*

$$\begin{array}{rcl} X,\ Y & : & U_0 \\ C & : & (U_1 \times U_1)\mathbf{Type} \\ f & : & (x, y : U_1)C(pair_\times(U_1, U_1, x, y)) \end{array}$$

and the following term M:

$$M \equiv \mathcal{E}_\times(U_1, U_1, C, f, pair_\times(U_0, U_0, X, Y))$$

where $\mathcal{E}_\times$ is the elimination operator for cartesian product types. M is well-typed (because $U_0 \times U_0 \leq U_1 \times U_1$) and of type

$$A \equiv C(pair_\times(U_0, U_0, X, Y)).$$

Now, $M \rhd f(X, Y) : C(pair_\times(U_1, U_1, X, Y))$. However, $f(X, Y)$ is not *of type A. Subject reduction fails to hold.*

Remark For some types, the elimination operators may take different forms than that for general inductive types. For example, in Coq, there is subsumptive subtyping on Π-types with application operators and cumulativity and, in ECC, the same for Π and there is also subsumptive subtyping for Σ-types with projection operators and the corresponding cumulativity. In such situations, things are OK (eg, the meta-theory goes through) because the elimination rules are not in the general form for inductive types. If the latter, the above problems would occur.

Adding Definitional Equality Rules: Discussion. One might say that the above problems are due to some sort of 'syntactic anomaly' and the syntax can be fixed. For instance, one might propose to ignore the typing information when equality/conversion testing is performed. More precisely, one may consider adding new definitional equality rules as guided by the definitions of the propagating coercions.[4] For instance, for $List(A)$, guided by equations (1) and (2) in §2, we might omit all of the typing/coercion information and impose the following definitional equalities:[5] for $A \leq B$, $a : A$ and $l : List(A)$,

$$\begin{array}{rcl} nil(A) & = & nil(B) \\ cons(A, a, l) & = & cons(B, a, l) \end{array}$$

The hope would be that, with the Russell-style universes and such additional definitional equality rules, the resulting type theory would still have nice properties such as canonicity and strong normalisation. However, besides that such a proposal is a bit inconsistent with the spirit of canonical objects, it is unclear whether it can go through: some substantial and tedious meta-theoretic work is called for, along the line of, for example, [Gog94, Wer94].

In the next section, we shall make a proposal to employ the Tarski-style universes together with coercive subtyping so that, we can achieve both of the theoretical correctness (eg, good meta-theoretic properties), on the one hand, and the ease-of-use in practice, on the other.

4 Tarski-style Universes with Coercive Subtyping

Our proposal is to employ the Tarski-style universes and to adopt the following coercion declarations and notational conventions so that the use of universes is as easy as the Russell-style universes, although 'behind the scenes' it is still the Tarski-style universes.

Lifting Operators as Coercions. The lifting operators $t_{i+1} : (U_i)U_{i+1}$ are taken as coercions. (Note that coercions are 'implicit' in the sense that they are not seen by the users.)

[4]A similar proposal was made by Dr Conor McBride in response to a talk of mine on the subject in the 2010 Fun forum in London and by Dr Bruno Barras in a recent communication: they have both suggested to impose similar equality rules.

[5]I am not precise here. Please note that even the formulation of such definitional equality rules may not be as simple as one might think.

Some Notational Conventions. We adopt the following conventions (to abbreviate the left-hand side by the right-hand side):

- $[El(A)] = [A]$ (El in the logical framework is omitted.)
- $[T_i(a)] = [a]$ (T_i is omitted.)
- Types and their names are (syntactically) 'identified': for example,
  - $[u_i] = U_i$ (U_i and u_i are identified.)
  - $[nat] = Nat$ (Nat and nat are identified.)
  - $[\sigma(a, b)] = \Sigma([a], (x{:}[a])[b(x)]$ ($\Sigma(A, [x{:}A]B(x))$ and $\sigma(a, b)$ are identified if a names A and $b(x)$ names $B(x)$.)

We claim that the above proposal achieves our goals, because

- whether a term denotes a type or its name can be automatically disambiguated from the context;
- with the above notational conventions, we obtain all the Russell-style rules except the rule 'if $A : U_i$ then $A : U_{i+1}$'; but
- this rule is taken care of by declaring the lifting operators t_i as coercions.

Remark Voevodsky [Voe12] has introduced a type system TS whose universes are indexed by level expressions which may contain level variables. In particular, if M_1 and M_2 are level expressions such that $M_1 \leq_{\mathcal{A}} M_2$, then there is a lifting operator j_{M_1,M_2} from U_{M_1} to U_{M_2}. The above proposal extends to the universes indexed by level expressions with:

- similar notational conventions;
- lifting operators j_{M_1,M_2} as coercions; plus
- the definitional equalities $j_{M_1,M_3} = j_{M_2,M_3} \circ j_{M_1,M_2}$ for $M_1 \leq_{\mathcal{A}} M_2 \leq_{\mathcal{A}} M_3$ (cf, TS's reduction rules).

References

[Gog94] H. Goguen. *A Typed Operational Semantics for Type Theory.* PhD thesis, University of Edinburgh, 1994.

[LSX12] Z. Luo, S. Soloviev, and T. Xue. Coercive subtyping: theory and implementation. *Information and Computation*, 2012. (To appear).

[Luo99] Z. Luo. Coercive subtyping. *Journal of Logic and Computation*, 9(1):105–130, 1999.

[ML84] P. Martin-Löf. *Intuitionistic Type Theory.* Bibliopolis, 1984.

[Pal98] E. Palmgren. On universes of type theory. In G. Sambin and J. Smith, editors, *Twenty-Five Years of Constructive Type Theory.* Oxford University Press, 1998.

[Voe12] V. Voevodsky. A universe polymorphic type system, 2012. (Version downloaded on October 8, 2012).

[Wer94] B. Werner. *Une Théorie des Constructions Inductives.* PhD thesis, 1994.