Universes in Type-Theoretical Semantics

Zhaohui Luo

Abstract  In type theory, a universe is a type of types and adding universes may increase the expression power and is very useful in applications. In this paper, we shall first consider how to model adjectival modification and coordination as examples to show how universes may be used to enable a form of polymorphism (Π-polymorphism) in semantic constructions. Then, we study a new kind of universes – subtype universes, which in particular allow one to use bounded quantification (quantification over subtypes) in the applications such as linguistic semantics. Introducing a form of universes, one is required to show that the extension still enjoys the nice properties of the original type theory including, for example, logical consistency and normalisation. We show that the extension with subtype universes does preserve these properties.

1 Introduction

In modern type theories (MTTs for short) a universe is a type of types. Put in another way, a universe is a type whose objects are also types. For instance, Martin-Löf introduced predicative universes $U_i$ (or called $Type_i$, $i \in \omega$) with the motivation of making the type system more expressive [24, 25] (e.g., in his predicative type theory, one can only prove that $0 \neq 1$ with a universe [31].) When MTTs are used as foundational languages for linguistic semantics [28, 21, 6], universes provide powerful means in formal semantics including, for example, the generic expressiveness in a form of polymorphism (called Π-polymorphism), which has proven to be very useful in various semantic constructions.

In this paper, we shall first consider how to model adjectival modification and coordination as examples to show how universes and associated Π-polymorphism can be used in semantic interpretations. Then, we study a new kind of universes – subtype
universes, which can be used in various ways, one of which is allow quantification over subtypes of a type in the form of $\forall X \leq H. P(X)$ which intuitively states that ‘for all subtypes $X$ of $H$, $P(X)$ holds’, called bounded quantification in the literature. Allowing such a quantification provides rather powerful means in various semantic constructions. We shall use examples in modelling subsective adjectives such as ‘skilful’ and gradable adjectives such as ‘tall’ to illustrate how subtype universes can be employed effectively in constructing semantic interpretations.

Introducing universes in type theory requires us to be very careful because it may have unintended consequences including, sometimes, even logical paradoxes, which would be disastrous for a foundational semantic language. For instance, bounded quantification in the calculus $F \leq$ was once shown to be undecidable [27] and, since subtype universes do introduce bounded quantification, is it OK for us to introduce subtype universes into MTTs? The answer to this question is positive, as recently shown by Maclean and the author [23]. Besides this positive result, we shall also briefly explain why introducing subtype universes into MTTs does not suffer from the undecidability problem as known for $F \leq$.

2 Universes and $\Pi$-polymorphism in Semantic Constructions

In general, universes are useful since they make it possible to treat various collections of types as internal totalities and, combined with $\Pi$-types, they help to represent concepts polymorphically or generically ($\Pi$-polymorphism). Typical examples of universes in MTT-semantics include $\text{CN}$, the universe of common nouns, and $\text{LType}$, the universe of ‘linguistic types’: we now use them as examples to explain how universes and the associated $\Pi$-polymorphism can be employed usefully in semantic constructions.

2.1 Universe $\text{CN}$ of Common Nouns

Introduced in [20, 21], $\text{CN}$ is the universe of all (interpretations of) common nouns. As such, $\text{CN}$ is the type whose objects include types such as $\text{Student}$ that interprets ‘student’ and $\Sigma x:\text{Cat}.\text{black}(x)$ (semantics of ‘black cat’), which is a subtype of $\text{Cat}$ (semantics of ‘cat’), and many other types that interpret common nouns. See Fig. 1 for a pictorial illustration.

$\text{CN}$ and the associated $\Pi$-polymorphic mechanism are very useful in semantic constructions. For example, it can be used to give semantics to verb-modifying adverbs [21], quantifiers [16] and subsective adjectives [5], as exemplified by (1),

\[1 \Sigma \text{ is a dependent type constructor. } \Sigma x:A.B(x) \text{ contains the dependent pairs } (a, b) \text{ such that } a \text{ is of type } A \text{ and } b \text{ is of type } B(a). \text{ In the coercive subtyping mechanism [19, 22] employed in MTT-semantics, we usually have } \Sigma x:A.B(x) \leq \pi_1 A, \text{ where coercion } \pi_1 \text{ is the first projection – see Appendix on } \Sigma\text{-types (at the end the paper after its references) for the inference rules for } \Sigma\text{-types.} \]
(2) and (3), respectively.

(1) \textit{quickly} : \Pi A : \text{CN}. (A \to \text{Prop}) \to (A \to \text{Prop})
(2) \textit{some} : \Pi A : \text{CN}. (A \to \text{Prop}) \to \text{Prop}
(3) \textit{small} : \Pi A : \text{CN}. (A \to \text{Prop})

All of the above examples use \emph{\Pi-poly}morphism over the universe \text{CN}. Let’s explain them in turn.

First, consider verb-modifying adverbs such as ‘quickly’, as exemplified by (1). In Montague semantics, one could give the semantic typing for a verb-modifying adverb like ‘quickly’ as of type in (4): it takes a predicate of type \(e \to t\) and returns a predicate of the same type.

(4) \textit{quickly}_{M} : (e \to t) \to (e \to t)

In Montague semantics, every predicate has the same type \(e \to t\) since there is only one domain \(e\) of entities. MTT-semantics is different where CNs are interpreted as types; we do not just have one type of entities, but many of them such as \textit{Student}, \textit{Cat}, etc. Then, what is the typing for a verb-modifying adverb such as ‘quickly’? Here, a type universe and the associated mechanism of \Pi-poly\-morphism become useful: for example, ‘quickly’ can be given the polymorphic typing (1) which quantifies over the universe \text{CN} of common nouns and, when applied to any common noun \(A\), quickly\((A) : (A \to \text{Prop}) \to (A \to \text{Prop})\). For instance, assuming \(\text{run} : \text{Human} \to \text{Prop}\), then ‘run quickly’ would be interpreted as quickly\((\text{Human}, \text{run})\) of type \text{Human} \to \text{Prop}.

Similarly, as exemplified by (2), quantifiers can be typed by means of \Pi-poly\-morphism over \text{CN} as well. As a simple example, the quantifier ‘some’ may be defined by means of the logical quantifier \(\exists\) as in (5), with \text{CN} as its restricted domain, and its typing is (2) above; i.e., unlike the logical quantifier \(\exists\) which can be applied to any type, \textit{some} can only be applied to a type that interprets a CN. For instance, (6) can be interpreted as (7), where speak : \text{Human} \to \text{Prop} with \textit{Student} \leq \text{Human}. In other words, \textit{some} can be applied to \textit{Student} as in (7). On
the other hand, *some* cannot be applied to a type that does not interpret a CN: for example, the type \( \text{Prop} \to \text{Student} \) does not interpret any CN and, therefore, (8) is ill-typed since \( \text{Prop} \to \text{Student} \) is not in the universe \( \text{CN} \).

(5) \[ \text{some} = \lambda A : \text{CN} \lambda P : A \to \text{Prop}. \exists (A, P). \]

(6) Some students spoke.

(7) \text{some(Student, speak)}

(8) \text{some(Prop} \to \text{Student, ...)}

Finally, subsective adjectives such as ‘small’, as exemplified by (3), can be interpreted by means of \( \Pi \)-polymorphism over \( \text{CN} \). They are different from intersective adjectives whose modification can be directly interpreted by means of \( \Sigma \)-types (dependent types of pairs, see Footnote 1), since the meaning of a subsective adjectival modification depends on the common nouns they modify. For instance, consider the adjective ‘small’. If one used (9) and (10) to interpret ‘small elephant’ and ‘small animal’, respectively, then one would incorrectly conclude that every small elephant is a small animal, as expressed by the subtyping relationship in (11), since we usually have \( \text{Elephant} \leq \text{Animal} \).

(9) \[ \Sigma x : \text{Elephant. SMALL}(x) \]

(10) \[ \Sigma x : \text{Animal. SMALL}(x) \]

(11) \text{(#)} \text{small elephant} \leq \text{small animal}

\( \Pi \)-polymorphism associated with \( \text{CN} \) gives us a simple solution. The idea is that the typing of a subsective adjective is polymorphic over \( \text{CN} \): for example, \text{small} has the polymorphic type (3) and, for each common noun \( A : \text{CN} \), we have a predicate \( \text{small}(A) : A \to \text{Prop} \). With this typing, ‘small elephant’ and ‘small animal’ are interpreted as (12) and (13), respectively. Here, \( \text{small(Elephant)} \) and \( \text{small(Animal)} \) are different predicates and we would not be able to deduce the incorrect (11) anymore.

(12) \[ \text{small elephant} = \Sigma x : \text{Elephant. small(Elephant, x)} \]

(13) \[ \text{small animal} = \Sigma x : \text{Animal. small(Animal, x)} \]

The above solution captures the intended meanings of subsective adjectives satisfactorily. It is basically an implementation of the intuition that the meaning of a subsective adjective depends on the particular CN it modifies in each case. Thus, a small elephant is only small with respect to elephants, a skilful surgeon is only skilful as a surgeon, and so on. Using the type proposed in (3), we can have different instances of a subsective adjective, say \( \text{small}(A) \), depending on the choice of common noun \( A : \text{CN} \).
2.2 Universe LType for Coordination

The universe \( LType \) was introduced to study coordination in MTT-semantics [4]. For instance, the word ‘and’ can be used to connect many entities of different types including, for example, sentences, verbs, adjectives, adverbs, quantified NPs, proper names, and many others, as exemplified by (14-20).

(14) John walks and Mary talks. (sentences)
(15) John walks and talks. (verbs)
(16) Mary is pretty and smart. (adjectives)
(17) The plant died slowly and agonizingly. (adverbs)
(18) Every student and some professor came. (quantified NPs)
(19) Some but not all students got an A. (quantifiers)
(20) John and Mary went to Italy. (proper names)

Semantically, people have called these different types conjoinable [26, 12]. In the framework of MTT-semantics, they correspond to a wide range of semantic types such as \( Prop \) (for sentences), \( A \to Prop \) (for verbs with domain \( A \)), \( \Pi A:CN. (A \to Prop) \to (A \to Prop) \) (for verb-modifying adverbs), and many others.

We introduce a universe \( LType \) that contains all these conjoinable types as its objects, including those types of predicates and CNs, among others. Then, we can use \( \Pi \)-polymorphism associated with \( LType \) to give a generic semantic typing of ‘and’ as in (21) and the coordination cases can all be dealt with generically.

(21) \( \text{And} : \Pi A:LType. A \to A \to A \)

For instance, (14) and (15) can now be given (22) and (23) as semantics, respectively. In these examples, the first argument of \( \text{And} \) is a conjoinable type – \( Prop \) in (22) and \( Human \to Prop \) in (23). This illustrates the use of the mechanism of \( \Pi \)-polymorphism in the typing of \( \text{And} \) in (21).

(22) \( \text{And}(Prop, \text{walk}(j), \text{talk}(m)) \)
(23) \( \text{And}(Human \to Prop, \text{walk}, \text{talk})(j) \)

The semantic operator \( \text{And}(A) \) can be defined as intended for each conjoinable type \( A \). For instance, for propositions of type \( Prop \), its meaning can be defined as (24) by which we’ll have (25), as intended.

(24) \( \text{And}(Prop, P, Q) = P \land Q \)
(25) \( \text{And}(Prop, \text{walk}(j), \text{talk}(m)) = \text{walk}(j) \land \text{talk}(m) \)

Formally, the introduction rules of \( LType \) are given in Fig. 2. \( LType \) contains the following kinds of types as its objects:

- the predicate types of the form \( \Pi x_1:A_1...\Pi x_n:A_n.Prop; \) and
In [4], we have not studied how to characterise the semantic behaviours such as (24-25) of the coordinating connectives. We now use \textit{And} as an example to describe how to define their behaviours: in the following cases of \(A\), the semantic behaviour of \(\text{And}(A): A \rightarrow A \rightarrow A\) is defined as follows:

- If \(A = \Pi_{x_1:A_1} \ldots \Pi_{x_n:A_n}.Prop\) then, for any \(f, g : A\),
  
  \[
  \text{And}(A, f, g)(x_1, \ldots, x_n) =_{\text{df}} f(x_1, \ldots, x_n) \land g(x_1, \ldots, x_n) : \text{Prop}.
  \]

  When \(n = 0\), it reduces to (24) above.
- If \(A : \text{CN}\) then, for any distributive predicate \(P : A \rightarrow \text{Prop}\),
  
  \[
  P(\text{And}(A, a_1, a_2)) =_{\text{df}} P(a_1) \land P(a_2).
  \]

With this description of the behaviour of \textit{And}, the examples (14-20) can be given semantics by \textit{And} as follows, where (26) is just repeating (25):

(26) (sentences)
\[
\begin{align*}
\text{[John walks and Mary talks]} &= \text{And}(Prop, walk(j), talk(m)) \\
&= \text{walk}(j) \land \text{talk}(m).
\end{align*}
\]

(27) (verbs)
\[
\begin{align*}
\text{[John walks and talks]} &= \text{And}(Human \rightarrow Prop, walk, talk)(j) \\
&= \text{walk}(j) \land \text{talk}(j).
\end{align*}
\]

(28) (adjectives)
\[
\begin{align*}
\text{[Mary is pretty and smart]} &= \text{And}(Human \rightarrow Prop, pretty, smart)(m) \\
&= \text{pretty}(m) \land \text{smart}(m).
\end{align*}
\]

(29) (adverbs)
\[
\begin{align*}
\text{[The plant died slowly and agonizingly]} &= \text{And}(T, slowly, agonizingly)(Plant, die, the\_plant) \\
&= \text{slowly}(Plant, die, the\_plant) \land \text{agonizingly}(Plant, die, the\_plant),
\end{align*}
\]
where \(T = \Pi A : \text{CN}.(A \rightarrow \text{Prop}) \rightarrow (A \rightarrow \text{Prop})\).

\(^2\) Note that there is no such a reductive equation for a verb whose semantics is a collective predicate, as illustrated by a sentence like “Whitehead and Russell wrote \textit{Principia Mathematica}.”
Universes in Type-Theoretical Semantics

(30) (quantified NPs)

\[
\left[ \text{Every student and some professors came} \right] = \text{And}(\text{Human} \rightarrow \text{Prop}) \rightarrow \text{Prop}, \text{every(\text{Student})}, \text{some(\text{Professor})})(\text{come}) = \forall(\text{Student}, \text{come}) \land \exists(\text{Professor}, \text{come}).
\]

(31) (quantifiers)

\[
\left[ \text{Some but not all students got an A} \right] = \text{And}(\Pi: \text{CN}.(A \rightarrow \text{Prop}) \rightarrow \text{Prop}, \text{some, notAll})(\text{Student}, \text{getA}) = \exists(\text{Student}, \text{getA}) \land \neg\forall(\text{Student}, \text{getA}).
\]

(32) (proper names)

\[
\left[ \text{John and Mary went to Italy} \right] = \text{goItaly}(\text{And}(\text{Human}, j, m)) = \text{goItaly}(j) \land \text{goItaly}(m).
\]

3 Subtype Universes

3.1 Subtype Universes and Bounded Quantification

A subtype universe is a type that represents a collection of subtypes: for any type \( H \), the universe \( U(H) \) represents the collection of all subtypes of \( H \). Such subtype universes can be specified formally by the following formation rule \( (U_F) \) and introduction rule \( (U_I) \), where \( A \leq H \) is the shorthand for \( 'A \leq c H \) for some coercion \( c' \) in the framework of coercive subtyping [18, 22]:

\[
\begin{align*}
(U_F) & \quad \Gamma \vdash H \text{ type} \quad \Gamma \vdash U(H) \text{ type} \\
(U_I) & \quad \Gamma \vdash A \leq H \quad \Gamma \vdash A : U(H)
\end{align*}
\]

Such type universes can be quantified over to form other propositions. This, among other things, gives a nice treatment of bounded quantification of the form

\[ \forall X \leq H. P(X) \]

Intuitively, it allows one to quantify over all subtypes of a type \( H \). This is exactly done by quantification over the subtype universe \( U(H) \): intuitively, the above formula with bounded quantification is the same as

\[ \forall X : U(H). P(X), \]

both saying that the property \( P \) holds for all subtypes of \( H \).

\[ \text{It is worth remarking that subsumptive subtyping, the traditional notion of subtyping where} \ A \leq B \ \text{means that all objects of type} \ A \ \text{are also of type} \ B, \ \text{is not adequate for a modern type theory because its introduction would make the type theory lose its important properties such as canonicity. See, for example, Section 4 of [22] for more details.} \]
The notion of bounded quantification was first proposed by Cardelli and Wegner in the programming language \textit{Fun} \cite{Fuss} (and in the context of typeful programming \cite{Fun}). Its theoretical properties have been extensively studied in the second-order systems such as \(F_\leq\) \cite{Fuss, Fun}.\footnote{The systems \textit{Fun} and \(F_\leq\) are all extensions of the second-order polymorphic \(\lambda\)-calculus (or the system \(F\)) \cite{Fuss, Fun}, which some researchers have employed to study formal semantics (see, for example, work by Retoré and his colleagues \cite{Retore}).} It was shown by Pierce \cite{Pierce} that type-checking in \(F_\leq\) is undecidable, which has significant impacts on further studies of bounded quantification (and hence its logic under the propositions-as-types principle would be problematic).\footnote{This negative result \cite{Pierce} had some unexpected implications: for example, it did mislead some people to think (mistakenly) that introducing bounded quantification into modern type theories might be problematic. Fortunately, this has been proven not to be the case.} However, as Maclean and the author have recently discovered \cite{Maclean}, introducing subtype universes as described by the above rules into modern type theories is indeed OK, preserving logical consistency, among other properties. In other words, it avoids the type checking problem traditionally associated with it (see Sect. \ref{sect:examples} below for further discussions about why this is the case, among other things).

\section{Examples in Linguistic Semantics}

Bounded quantification is very useful in many applications including modelling programming techniques and formal semantic interpretations. To illustrate their use in the latter, we consider two examples in this section.

\textbf{Gradable Adjectives}. We consider how to give semantic interpretations of gradable adjectives with subtype universes. Gradable adjectives (words such as ‘tall’) can be interpreted as predicates which involve comparison of a measurement on the entity with some threshold value, and their semantic modelling provides a challenging case for MTT-semantics without subtype universes \cite{Gradable} (and also see the references there about the studies of gradable adjectives in the Montagovian setting). In the case of ‘tall’, the measurement is the height of the argument. Furthermore, the threshold often varies based on the type of the argument. The threshold height that is considered tall for a human is very different from the height considered tall for a building and, furthermore, it can also be different from that for boys, although all boys are humans. We now describe how to use subtype universes in the formal modelling of its semantics.

Let \(V_h\) be a type universe whose objects are base types such as \textit{Human} and \textit{Building} for which the following makes sense:

\[\text{height} : \Pi H : V_h. (H \rightarrow \text{Nat}),\]
where $Nat$ is the type of natural numbers and, for example, $\text{height}(\text{Human}, \text{John}) = 175$ would mean that John’s height is $1.75m$. Then, the type of tall can be given by means of subtype universes as in (33), which can be rewritten as (34) in the notation for bounded quantification:

(33) $\text{tall} : \Pi H : V_h \Pi A : U(H), (A \to \text{Prop})$

(34) $\text{tall} : \Pi H : V_h \Pi A \leq H, (A \to \text{Prop})$

So, tall is a predicate on subtypes of the base types. For instance, if $\text{Human} : V_h$ and $\text{socrates} : \text{Man} \leq \text{Human}$, then $\text{tall}(\text{Human}, \text{Man}, \text{socrates})$ is a proposition. Given a threshold function

$$\xi : \Pi H : V_h, (U(H) \to Nat),$$

one may define tall as follows:

(35) $\text{tall}(H, A, x) = \text{height}(H, x) \geq \xi(H, A)$

Note that $\text{John} : \text{Man}$ being tall (as a man) and $\text{Oliver} : \text{Boy}$ being tall (as a boy) can be rather different and the difference is reflected in that the threshold values $\xi(\text{Human}, \text{Man})$ and $\xi(\text{Human}, \text{Boy})$ are different.

Multidimensional adjectives such as ‘healthy’ can also be modelled by means of subtype universes. For example, ‘healthy’ may be given the type (36) which can be rewritten as (37) in bounded quantification.

(36) $\text{Healthy} : \Pi A : U(\text{Human}), (A \to \text{Prop})$

(37) $\text{Healthy} : \Pi A \leq \text{Human}, (A \to \text{Prop})$

With healthy thresholds $\xi_i : \Pi A \leq \text{Human}.Nat$ with indexes $i$ such as $BP$ (for blood pressure), we have, for $A \leq \text{Human}$, $\text{Healthy}(A, x) = \bigvee_i \chi_i(A, x)$, where $\chi_i$’s are the corresponding propositions: for instance, $\chi_{BP}(A, x) = BP(x) \leq \xi_{BP}(A)$, where $A$ is a subtype of Human, examples of which include Boy and Woman.

**Skilful: a subsective adjective.** Let’s consider another example: how to interpret the subsective adjective ‘skilful’. In Sect. 2.1, it is shown how to use the universe $\text{CN}$ to interpret subsective adjectives like ‘small’ whose type would be (3), repeated here as (38):

(38) $\text{small} : \Pi A : \text{CN}, (A \to \text{Prop})$

Can we do the same for ‘skilful’, as in (39)?

(39) $(\#) \text{skilful} : \Pi A : \text{CN}, (A \to \text{Prop})$

Unfortunately, this is not quite adequate. Yes, if we apply skilful to Doctor, then we get a predicate $\text{skilful}(\text{Doctor}) : \text{Doctor} \to \text{Prop}$, as expected. However, one can also have, for example, $\text{skilful}(\text{Building}) : \text{Building} \to \text{Prop}$, which is
obviously not intended. Such unintended combinations should usually be excluded: the question is how?

With subtype universes, what we can do is to introduce a universe $\text{CN}_H$ which contains those common nouns that are subtypes of $\text{Human}$ (that’s what the subscript $H$ in $\text{CN}_H$ stands for). In other words, we have the following introduction rule for the subtype universe $\text{CN}_H$:

$$
\begin{align*}
A &: \text{CN} \\
A &\leq \text{Human} \\
\implies \\
A &: \text{CN}_H
\end{align*}
$$

Then, ‘skilful’ can be given semantic typing as (40):

$$(40) \text{skilful} : \prod A : \text{CN}_H. (A \rightarrow \text{Prop})$$

We can of course still have the expected cases like $\text{skilful}(\text{Doctor})$ because $\text{Doctor} \leq \text{Human}$, and we have now excluded the unintended cases such as $\text{skilful}(\text{Building})$ because $\text{Building} \not\leq \text{Human}$ and hence $\text{skilful}(\text{Building})$ is ill-typed. With subtype universe $\text{CN}_H$, we claim that (40) is an adequate typing for $\text{skilful}$ and subtype universes have given us a good solution here.

### 3.3 Meta-Theoretic Studies of Subtype Universes

Introducing universes into type theories requires us to be very careful because it may have some unexpected consequences including potentially resulting in logical paradoxes. A typical example is to introduce a universe Type of all types, including itself ($\text{Type} : \text{Type}$). It is known that the introduction of Type (together with the $\Pi$-types) would result in Girard’s paradox [11, 8, 13] in that every type becomes inhabited (and, hence, the type theory is logically inconsistent under the propositions-as-types principle [10, 14]).

Fortunately, as Maclean and the author have recently discovered, introducing subtype universes into modern type theories, as described by the rules $(U_F)$ and $(U_I)$ in the last subsection, is indeed OK in the sense that it preserves nice properties such as logical consistency and strong normalisation, as informally stated in the following theorem. A key idea of its proof is to stratify the subtype universes, mapping the type theory to one with ‘predicative universes’ $\text{Type}_i$ ($i \in \omega$) such as $\text{UTT}$ [17]: for instance, if $H$ is mapped to a type in $\text{Type}_1$, then $U(H)$ is mapped to a type in $\text{Type}_{i+1}$, $U(U(H))$ to a type in $\text{Type}_{i+2}$, etc. The detail of such a proof can be found in [23].

**Theorem** The addition of subtype universes to a type theory preserves its nice properties such as logical consistency, strong normalisation, and decidability of type-checking.

As mentioned in Sect. 3.1, quantification over a subtype universe is intuitively equivalent to bounded quantification, therefore, introducing subtype universes is
related to the study of bounded quantification, especially as we know that the system $F_\leq$ is undecidable [27]. One would expect us to explain the difference between introducing subtype universes to MTTs and introducing bounded quantification in $F_\leq$ and, in particular, why it is the case that the former is OK, while the latter is not. We have analysed this and found out that a key problem for the undecidability of $F_\leq$ is the existence of the top type $\text{Top}$ in $F_\leq$: $\text{Top}$ is the top type in $F_\leq$ in the sense that, for every type $A$, $A \leq \text{Top}$.\footnote{$\text{Top}$ was introduced by Cardelli and Wegner in their language $\text{Fun}$ [3]. Its main purpose was to recover the ‘ordinary’ unbounded quantification $\forall X.P(X)$ as a special case of bounded quantification by defining it to be $\forall X \leq \text{Top}.P(X)$.} It is indeed the presence of this top type in $F_\leq$ that has caused its undecidability. Analysed in this way, this is not surprising anymore: the existence of $\text{Top}$ is actually allowing one to quantify over all types, including $\text{Top}$ itself. You may want to compare this with the type $\text{Type}$ of all types mentioned above: the bounded quantification $\forall \alpha \leq \text{Top}$ is effectively quantifying over all types, something very close to, if not exactly the same as, having a type of all types for which one can quantify over. Indeed, as shown by Katiyar and Sankar [15], removing the top type $\text{Top}$ from $F_\leq$, the resulting calculus would become decidable. MTTs do not have a top type (and, in particular, it does not have a type of all types – see above), therefore, the positive result as stated in the above theorem is not that surprising after all, as compared with the undecidability result for $F_\leq$.$^7$

4 Concluding Remarks

We have discussed how universes can be employed in type-theoretical semantics, using the universes $\text{CN}$ and $\text{LType}$ as examples for illustration. Introducing a new kind of universes $U(H)$, called subtype universes, we showed that they can extend the expressive power in a useful way for semantic constructions, and explained that their introduction in type theory does preserve the nice properties such as logical consistency and decidability of type-checking.

There are both theoretical and practical issues to be studied for subtype universes. For linguistic semantics, we need to make use of subtype universes in more case studies to judge their effectiveness, on the one hand, and their weakness (if any), on the other. Theoretically, it would be interesting to investigate the relationship between subtype universes $U(H)$ and the ‘existing’ predicative universes $\text{Type}_i$. For instance, in our proof of the theorem in Sect. 3.3, we have $U(H) : \text{Type}_{i+1}$ if $H : \text{Type}_i$. This has intuitively assumed that we would have a problem (say, inconsistency) if we had, for example, $U(\text{Type}(0)) : \text{Type}(0)$. More detailed investigation for such relationships is called for.

\footnote{Of course, the proof of the theorem [23] involves other intricacies, but as far as the comparison with the undecidability of $F_\leq$ is concerned, the analysis has given us a straightforward explanation.}
References

25. Pierce, B.: Bounded quantification is undecidable. Information and Computation 112(1)
Appendix on Σ-types

The Σ-type $\Sigma x : A. B(x)$ consists of pairs $(a, b)$ such that $a$ is of type $A$ and $b$ is of type $B(a)$. Σ-types are associated with the projection operators $\pi_1$ and $\pi_2$ so that, for $(a, b)$ of type $\Sigma x : A. B(x)$, $\pi_1(a, b) = a$ and $\pi_2(a, b) = b$. Formally, Σ-types are governed by the following inference rules.

\[
\frac{\text{A type} \quad \text{B type} \quad \{x : A\}}{\Sigma x : A. B \text{ type}}
\]

\[
\frac{a : A \quad b : \{a/x\}B \quad \text{B type} \quad \{x : A\}}{(a, b) : \Sigma x : A. B}
\]

\[
\frac{p : \Sigma x : A. B \quad p : \Sigma x : A. B}{\pi_1(p) : A \quad \pi_2(p) : \{\pi_1(p)/x\}B}
\]

\[
\frac{a : A \quad b : \{a/x\}B \quad \text{B type} \quad \{x : A\}}{\pi_1(a, b) = a : A}
\]

\[
\frac{a : A \quad b : \{a/x\}B \quad \text{B type} \quad \{x : A\}}{\pi_2(a, b) = b : \{a/x\}B}
\]