On the Metatheory of Subtype Universes

Felix Bradley and Zhaohui Luo

Royal Holloway, University of London, Egham, U.K.
Felix.Bradley@rhul.ac.uk
zhaohui.luo@hotmail.co.uk

Introduction. Power types were first introduced in a seminal paper by Cardelli and formulated the notion of a 'type of subtypes', analogous to that of a power set in set theory. [2] This not only explicitly mechanised bounded quantification, a form of polymorphism wherein the type quantifier is restricted to only subtypes of a given type, but it also served as a method for both the type theory to talk about its own subtyping, and for the subtyping relation to be completed subsumed by the typing relation (by considering $A \leq B$ as a shorthand for $A : \text{Power}(B)$. The type theory in Cardelli's work promoted more powerful expression over more well-behaved metatheoretical properties, such as including **Type** : **Type**, which induces logical inconsistency and non-normalisation [4, 5].

Subtype universes were initially introduced by Maclean and Luo as a way of formalising the notion of a 'type of subtypes' for a well-studied logical type theory equipped with coercive subtyping [9]. This extended type theory has several nice metatheoretical properties such as strong normalisation, but this implementation excluded certain kinds of subtyping relations from being used, and the formulation was wrapped up in the complexities of the underlying type theory's universe hierarchy. In particular, the implementation excludes (in layman's terms) subtyping relations where there are any occurrences of $\mathcal{U}$, or wherein the supertype inhabits a type universe of a smaller level than that of the subtype.

We consider a simpler yet more expressive reformulation of subtype universes by amending a logical type theory equipped with coercive subtyping with the following rules

$$\frac{\Gamma \vdash B\,\text{type}}{\Gamma \vdash \mathcal{U}(B)\,\text{type}} \qquad \frac{\Gamma \vdash A \leq_c B}{\Gamma \vdash \langle A, c\rangle : \mathcal{U}(B)}$$

and two operators σ_1 and σ_2 which respectively extract the type and coercion of a subtype universe's 'pair'.

Expressive Subtype Universes. Prior work on subtype universes included the restriction that a subtype must inhabit a 'smaller' type universe than that of the supertype, and likewise must not include subtype universes. This was necessary in Maclean and Luo's work on extending UTT[$\mathcal{C}$] with subtype universes as their proof of logical consistency and strong normalisation was via an embedding of their extended type theory back into UTT[$\mathcal{C}$], primarily due to the preexisting hierachy of type universes through which UTT controls its predicative type structure.

This is not necessary in general, however. By considering extending a theory which only possesses an impredicative type universe of propositions Prop, such a type theory has the ability to use higher-level types on either side of the subtyping relation, such as $\mathcal{U}(A) \leq_c A$. Combined with the new operator that extracts a given coercion from a subtyping relation, these new features significantly expand on expressiveness of the type theory, allowing for more subtyping relations than Maclean and Luo's previous implementation.

There still remains the difficulty of proving that a collection of subtyping judgements and inference rules are coherent - "that every possible derivation of a statement $\Gamma \vdash a : A$ has the same meaning." [10] The difficulty of this task is increased for subtype universes when we consider subtyping judgements that include subtype universes, as the coherency of these judgements may now be dependent on other subtyping judgements.

Applications. Subtype universes, in a similar vein to power types, can be used to explicitly mechanise bounded quantification by considering $\lambda(X \leq A).M$ as syntactic sugar for $\lambda(x : \mathcal{U}(A)).[\sigma_1(x)/X]M$, and as such have a variety of uses in modelling programming languages. Additionally, the ability to extract coercions allows for more complex subtyping judgements and inference rules, allowing for a more rich type theory. This has been particularly insightful in formalisation of mathematics, wherein subtype universes have been rudimentarily useful in modelling the topologies of various spaces.

In particular, these additions have proven useful in natural language semantics, wherein types can be interpreted as common nouns and terms as specific instances of these nouns. Here, subtype universes can be used to model subsective adjectives. If CN is the universe of common nouns, then where previously we may have had a term $\text{skillful} : \Pi(A : \text{CN}).A \to \text{Prop}$, we may wish to exclude instances such as skillful(chair). By instead considering $\text{skillful} : \Pi(x : \mathcal{U}(\text{Human})).\sigma_1(x) \to \text{Prop}$ instead, we preserve desired terms such as $\text{skillful}(\langle \text{doctor}, c\rangle)$, and exclude undesired terms such as $\text{skillful}(\langle \text{chair}, c'\rangle)$.

Metatheory. We consider a subset τ of $\text{UTT}[\mathcal{C}]$ with some basic types such as $\mathbf{0}$, $\mathbf{1}$ and $\mathbb{N}$, dependent pair types and dependent function types, and no type universes other than Prop.. We have shown that extending τ with any collection of coherent subtyping judgements $\mathcal{C}$ preserves any underlying strong normalisation and logical consistency. This was done by considering an embedding of $\tau[\mathcal{C}]$ into its parent system $\text{UTT}[\mathcal{C}]$, which has been proven to be strongly normalising and logically consistent [7, 8]. The key idea is that subtype universes $\mathcal{U}(B)$ in our type theory should correspond to types of the form $\Sigma(X : \text{Type}_i).X \to B$ in $\text{UTT}[\mathcal{C}]$.

Describing this embedding is relatively easy when the subtype in a subtyping rule includes fewer uses of $\mathcal{U}$ than the supertype. We call subtyping judgements which satisfy this property 'monotonic', and show that a type theory with subtype universes extended only by coherent and monotonic $\mathcal{C}$ can trivially be embedded into $\text{UTT}[\mathcal{C}]$. For a subset of non-monotonic subtyping rules, we can alter our embedding - if the difference in the number of appearances of $\mathcal{U}$ on either side of a subtyping rule is always bounded by some k, then we can instead 'shift' our embedding by sending $\mathcal{U}(B)$ to $\Sigma(X : \text{Type}_{i+k}).X \to B$ instead.

Conclusion. Our work on generalising and extending subtype universes has allowed for a greater variety of subtyping relations while preserving strong normalisation and logical consistency. There are some open questions we wish to explore further: for example, including a universal supertype **Top** often presents metatheoretic issues, [3] [10] especially in conjunction with coercive subtyping, but it may be possible to develop structured or predicative alternatives which do not.

Our work has also explored models of point-set topology using subtype universes, interpreting $\mathcal{U}(B)$ as the topology of the space B: we would like to further refine this idea, and analyse the role that coercive subtyping plays in this. Finally, Aspinall's work on λ_{Power}, a predicative type theory using power types, has proven fruitful with results such as strong normalisation but has run into difficulties with certain metatheoretical proofs [1]. Hutchins' work on pure subtype systems generalises several subtyping to subsume typing entirely and has proven a very powerful system impredicative system without strong normalisation but with similar difficulties in the metatheory [6]. A better understanding of the complexities of these systems will be key going forward.

References

[1] David Aspinall. Subtyping with power types. In Peter G. Clote and Helmut Schwichtenberg, editors, *Computer Science Logic*, pages 156–171, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg.

[2] Luca Cardelli. Structural subtyping and the notion of power type. In *Proceedings of the 15th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '88, page 70–79, New York, NY, USA, 1988. Association for Computing Machinery. doi:10.1145/73560.73566.

[3] Adriana Compagnoni. Higher-order subtyping and its decidability. *Information and Computation*, 191(1):41–103, 2004. URL: https://www.sciencedirect.com/science/article/pii/S0890540104000094, doi:https://doi.org/10.1016/j.ic.2004.01.001.

[4] Thierry Coquand. An analysis of Girard's paradox. Technical Report RR-0531, INRIA, May 1986. URL: https://hal.inria.fr/inria-00076023.

[5] James G. Hook and Douglas J. Howe. Impredicative strong existential equivalent to type : Type. Technical report, Cornell University, Ithaca, New York, USA, June 1986.

[6] DeLesley S. Hutchins. Pure subtype systems. In *Proceedings of the 37th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '10, page 287–298, New York, NY, USA, 2010. Association for Computing Machinery. doi:10.1145/1706299.1706334.

[7] Zhaohui Luo. Coercive subtyping in type theory. In Dirk van Dalen and Marc Bezem, editors, *Computer Science Logic*, pages 275–296, Berlin, Heidelberg, 1997. Springer Berlin Heidelberg.

[8] Zhaohui Luo, Sergei Soloviev, and Tao Xue. Coercive subtyping: Theory and implementation. *Information and Computation*, 223:18–42, 2013. URL: https://www.sciencedirect.com/science/article/pii/S0890540112001757, doi:10.1016/j.ic.2012.10.020.

[9] Harry Maclean and Zhaohui Luo. Subtype Universes. In Ugo de'Liguoro, Stefano Berardi, and Thorsten Altenkirch, editors, *26th International Conference on Types for Proofs and Programs (TYPES 2020)*, volume 188 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 9:1–9:16, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. URL: https://drops.dagstuhl.de/opus/volltexte/2021/13888, doi:10.4230/LIPIcs.TYPES.2020.9.

[10] Benjamin C. Pierce. Bounded quantification is undecidable. In *Proceedings of the 19th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '92, page 305–315, New York, NY, USA, 1992. Association for Computing Machinery. doi:10.1145/143165.143228.