Commuting Rules for the Later Modality and Quantifiers in Step-Indexed Logics

Bálint Kocsis and Robbert Krebbers

Radboud University Nijmegen, The Netherlands
balint.kocsis@ru.nl mail@robbertkrebbers.nl

Abstract

Step-indexing is a semantic tool for stratifying circular, non-wellfounded definitions. The main idea is to use sequences of successive approximations to construct objects such as types, propositions, and functions. It can be formalised in a logical, or type-theoretical, setting, through a modality, called *later*, which allows us to talk about the next approximation.

The internal logic of the topos of trees provides a full-fledged higher-order logic with dependent types and built-in step-indexing. As in any modal logic, it is desirable to have commuting rules for modalities and quantifiers. However, the current known such rules are not satisfactory since they depend on an assumption on the domain of quantification.

We propose novel rules as alternatives from which the former ones can be derived. Our insights are based on the observation that the later modality can be decomposed into two parts. We have formalised our results in the Rocq Prover.

Step-indexing and the topos of trees. Step-indexing is a widely used semantic tool for stratifying circular, non-wellfounded definitions [2], particularly in program logics (*e.g.,* Iris [11]) and logical relation models of type systems with general recursive types [2] and dynamically allocated higher-order references [1]. Noteworthy recent applications of step-indexing involve a logical relation model of Rust [10] and a program logic based on session types [9].

The key idea of step-indexing is to use sequences of successive approximations to construct objects such as types, propositions, and functions, where the n-th approximation describes the object under the assumption that we only have n available computation steps to reason about it. To avoid tedious and low-level reasoning about step-indexed arithmetic, the *logical approach* to step-indexing [3, 8] employs the the *later* modality [12] to provide an abstraction.

The internal logic of the topos of trees [4] provides a full-fledged higher-order logic with dependent types, extended with a later modality on propositions ($\triangleright$) and a later modality on types ($\blacktriangleright$). This rich logic is a good candidate for defining program logics and logical relation models for complex programming languages internally.

Problem statement. Ideally, a logic should have neat and general rules for describing the interaction of the connectives. Unfortunately, the naive rules for commuting the later modality with quantifiers are unsound. Take the naive rule for commuting existentials and later:

$$\triangleright(\exists x : A.\, P) \dashv\vdash \exists x : A.\, \triangleright P$$

While the right-to-left direction follows from basic rules involving $\exists$ and $\triangleright$, the left-to-right direction does not hold because we quantify over different approximations of the domain A.

The current solution is to demand that the domain A is *total and inhabited* [4, 6]. This property can be expressed in the internal logic of the topos of trees as

$$\mathtt{TI}(A) := \forall y : \blacktriangleright A.\, \exists x : A.\, \mathtt{next}\, x = y.$$

Then we get the following inference rule:

$$\triangleright\text{-}\exists\text{-}\mathtt{TI}\quad \frac{\vdash \mathtt{TI}(A) \qquad \Gamma, x : A \vdash P : \mathtt{Prop}}{\Gamma \mid \triangleright(\exists x : A.\, P) \vdash \exists x : A.\, \triangleright P}$$

This rule is not ideal for two reasons. Firstly, it only works for total and inhabited types. Secondly, the assumption $\mathrm{TI}(A)$ has to be proved valid without any hypotheses. The second issue can be addressed by considering the following modified inference rule:

$$\triangleright\text{-}\exists\text{-}\mathtt{TI}\text{-}2\quad \frac{\Gamma, x : A \vdash P : \mathtt{Prop}}{\Gamma \mid \mathtt{TI}(A) \wedge \triangleright(\exists x : A.\, P) \vdash \exists x : A.\, \triangleright P}$$

This rule is also sound in the topos of trees, and it implies $\triangleright$-$\exists$-TI. However, the restriction on the domain of quantification remains.

Solution. Our proposed solution rests on the observation (first made by [4]) that, semantically speaking, the later modality $\triangleright : \mathtt{Prop} \to \mathtt{Prop}$ can be decomposed as $\triangleright = \mathtt{lift} \circ \mathtt{next}$, where $\mathtt{next} : \mathtt{Prop} \to \blacktriangleright\mathtt{Prop}$ and $\mathtt{lift} : \blacktriangleright\mathtt{Prop} \to \mathtt{Prop}$ (here, Prop is the type of propositions, *i.e.,* the subobject classifier in the topos of trees). Thus, we can study the interaction of the quantifiers with next and lift separately.

It follows from the homomorphism property of applicative functors that next commutes with the quantifiers, in the sense that we have equalities

$$\mathtt{next}\,(\exists x : A.\, Px) = \mathtt{next}\,\mathtt{ex} \circledast \mathtt{next}\,P \quad \text{and} \quad \mathtt{next}\,(\forall x : A.\, Px) = \mathtt{next}\,\mathtt{all} \circledast \mathtt{next}\,P,$$

where $\mathtt{ex}, \mathtt{all} : (A \to \mathtt{Prop}) \to \mathtt{Prop}$ are defined as:

$$\mathtt{ex}(P) = \exists x : A.\, Px \quad \text{and} \quad \mathtt{all}(P) = \forall x : A.\, Px.$$

Our contributions are the following two dual rules that relate lift with the quantifiers:

$$\mathtt{lift}\text{-}\exists\quad \frac{\Gamma \vdash Q : \blacktriangleright(A \to \mathtt{Prop})}{\Gamma \mid \mathtt{lift}\,(\mathtt{next}\,\mathtt{ex} \circledast Q) \dashv\vdash \exists y : \blacktriangleright A.\, \mathtt{lift}\,(Q \circledast y)}$$

$$\mathtt{lift}\text{-}\forall\quad \frac{\Gamma \vdash Q : \blacktriangleright(A \to \mathtt{Prop})}{\Gamma \mid \mathtt{lift}\,(\mathtt{next}\,\mathtt{all} \circledast Q) \dashv\vdash \forall y : \blacktriangleright A.\, \mathtt{lift}\,(Q \circledast y)}$$

It can be shown that lift-$\exists$ implies $\triangleright$-$\exists$-TI-2 and thus also $\triangleright$-$\exists$-TI. Hence, our new rules really are generalisations of the previously studied rules. This can be seen as an argument for the claim that lift is more primitive than $\triangleright$, and that step-indexed logics should focus on axiomatising the former.

Conclusion. We have found new and more general reasoning rules in the topos of trees for commuting the later modality with quantifiers. We proved the soundness of our new rules and showed that previously known rules can be derived from them. To ensure confidence in our results, we have formalised the topos of trees and its internal logic in the Rocq Prover [7].

It remains to be investigated how to apply our rules in practice. For instance, we would like to define a model of a step-indexed program logic (such as Iris [11]) in the internal logic of the topos of trees and see if our new rules are of use. Furthermore, one should investigate the interaction of `lift` with the other connectives in order to get a more complete axiomatisation of the internal logic.

We note that an operation similar to `lift`, called $\widehat{\triangleright}$, has already been studied in the context of guarded dependent type theory [5]. This operation is used to turn a code a for a type A to a code $\widehat{\triangleright}\, a$ for the type $\blacktriangleright A$. Under the Curry-Howard correspondence, propositions are expressed as types in dependent type theory, and thus, $\widehat{\triangleright}$ can be seen as a generalised version of `lift`. In such a setting, the rule `lift-∃` corresponds to the principle that the later type former preserves sigma types.

Finally, it would be worthwhile to investigate whether principles similar to `lift-∃` and `lift-∀` also hold in other models of step-indexing. In particular, it is conceivable that these rules are also valid in models where step-indexing is done over an ordinal larger than ω, such as Transfinite Iris [13].

Acknowledgments. We thank the anonymous reviewers for their helpful feedback.

References

[1] Amal J. Ahmed. *Semantics of types for mutable state*. PhD thesis, Princeton University, 2004.

[2] Andrew W. Appel and David A. McAllester. An indexed model of recursive types for foundational proof-carrying code. *ACM Trans. Program. Lang. Syst.*, 23(5):657–683, 2001. doi:10.1145/504709.504712.

[3] Andrew W. Appel, Paul-André Melliès, Christopher D. Richards, and Jérôme Vouillon. A very modal model of a modern, major, general type system. In *POPL*, pages 109–122, 2007. doi:10.1145/1190216.1190235.

[4] Lars Birkedal, Rasmus Ejlers Møgelberg, Jan Schwinghammer, and Kristian Støvring. First steps in synthetic guarded domain theory: step-indexing in the topos of trees. *Log. Methods Comput. Sci.*, 8(4), 2012. doi:10.2168/LMCS-8(4:1)2012.

[5] Ales Bizjak, Hans Bugge Grathwohl, Ranald Clouston, Rasmus Ejlers Møgelberg, and Lars Birkedal. Guarded dependent type theory with coinductive types. In *FoSSaCS*, volume 9634 of *LNCS*, pages 20–35, 2016. doi:10.1007/978-3-662-49630-5_2.

[6] Ranald Clouston, Ales Bizjak, Hans Bugge Grathwohl, and Lars Birkedal. The guarded lambda-calculus: Programming and reasoning with guarded recursion for coinductive types. *Log. Methods Comput. Sci.*, 12(3), 2016. doi:10.2168/LMCS-12(3:7)2016.

[7] The Rocq development team. The Rocq prover, 2025. URL `https://rocq-prover.org/`.

[8] Derek Dreyer, Amal Ahmed, and Lars Birkedal. Logical step-indexed logical relations. *Log. Methods Comput. Sci.*, 7(2), 2011. doi:10.2168/LMCS-7(2:16)2011.

[9] Jonas Kastberg Hinrichsen, Jesper Bengtson, and Robbert Krebbers. Actris 2.0: Asynchronous session-type based reasoning in separation logic. *Log. Methods Comput. Sci.*, 18 (2), 2022. doi:10.46298/LMCS-18(2:16)2022.

[10] Ralf Jung, Jacques-Henri Jourdan, Robbert Krebbers, and Derek Dreyer. Rustbelt: securing the foundations of the rust programming language. *Proc. ACM Program. Lang.*, 2(POPL): 66:1–66:34, 2018. doi:10.1145/3158154.

[11] Ralf Jung, Robbert Krebbers, Jacques-Henri Jourdan, Ales Bizjak, Lars Birkedal, and Derek Dreyer. Iris from the ground up: A modular foundation for higher-order concurrent separation logic. *J. Funct. Program.*, 28:e20, 2018. doi:10.1017/S0956796818000151.

[12] Hiroshi Nakano. A modality for recursion. In *LICS*, pages 255–266, 2000. doi:10.1109/LICS.2000.855774.

[13] Simon Spies, Lennard Gäher, Daniel Gratzer, Joseph Tassarotti, Robbert Krebbers, Derek Dreyer, and Lars Birkedal. Transfinite Iris: Resolving an existential dilemma of step-indexed separation logic. In *PLDI*, pages 80–95, 2021. doi:10.1145/3453483.3454031.