Substitution for Non-Wellfounded Syntax with Binders through Monoidal Categories

Ralph Matthes
IRIT, Université de Toulouse, CNRS, Toulouse INP, UT3, Toulouse, France

Kobe Wullaert
Delft University of Technology, Netherlands

Benedikt Ahrens
Delft University of Technology, Netherlands
University of Birmingham, United Kingdom

The present abstract is a condensed version of the preprint authored by the same authors [12].

1 General motivation for non-wellfounded syntax with binders

Non-wellfounded syntax with binders appears in its purest form in the coinductive reading of untyped λ-calculus. *Potentially* non-wellfounded λ-terms still consist of variables, λ-abstractions and applications only, but the construction process with these constructors can go on forever. Such construction processes can be described through functional programming, and the host programming language then serves as a meta-language for the description of those infinitary λ-terms. Instead of taking a programming perspective, one can also ask if a possibly circular definition of such a non-wellfounded term is well-formed, in the sense that it uniquely determines such a structure. Naturally, uniqueness is understood up to bisimilarity, i. e., two such non-wellfounded λ-terms are considered equal if their infinite unfoldings have the same labels (indicating the applied constructor) in the same order on each level, starting at the root. The presence of variable binding presents the extra challenge of having to consider this bisimilarity modulo renaming of bound variables, i. e., α-equivalence – a challenge that is amplified by the possibility of having an infinite number of bindings in a non-wellfounded λ-term. In this paper, we either work on an abstract level that does not reveal this challenge, or we resort to a representation using nested datatypes that is a form of de Bruijn representation with well-scopedness guaranteed by the typing system (see, e. g., [2]), and therefore α-equivalence is just not needed.

There are many uses of coinductive untyped λ-terms (such as Böhm trees), and coinductive readings of term structures with binding have a counterpart in infinitary rewriting.

2 A motivational application scenario

We have an application scenario in mind without the aforementioned "dynamics" of infinitary rewriting. Instead, the well-typed syntax itself is important and is more complicated than just λ-calculus, notably by the presence of embedded inductive types. This in particular motivates our search for *datatype-generic* constructions for a wide range of non-wellfounded simply-typed syntax.

The application scenario is as follows: We want to represent the entire search space for inhabitants in simply-typed λ-calculus (including infinite runs in a naive search loop) by a potentially non-wellfounded term of a suitable calculus (of the same type as the one for which we search inhabitants). This calculus is informally given by the following grammar:

Submitted to:
FICS 2024

$$\frac{\Gamma, x : A \vdash N : B}{\Gamma \vdash \lambda x^A.N : A \to B} \qquad \frac{(x : \vec{B} \to p) \in \Gamma \quad \forall i, \Gamma \vdash N_i : B_i}{\Gamma \vdash x\langle N_i\rangle_i : p} \qquad \frac{\forall i, \Gamma \vdash E_i : p}{\Gamma \vdash \sum_i E_i : p}$$

Figure 1: Coinductive typing rules for simple types in the application scenario

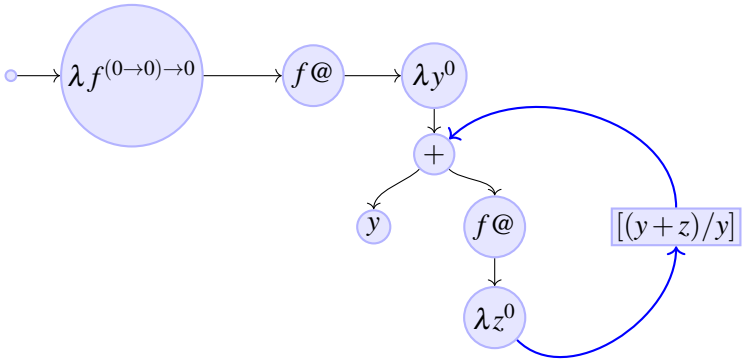

Figure 2: Forest representation of all inhabitants of THREE

$$\begin{array}{lll} \text{(terms)} & N & ::=_{co} \quad \lambda x^A.N \,|\, E_1 + \cdots + E_n \\ \text{(elimination alternatives)} & E & ::=_{co} \quad x\langle N_1, \ldots, N_k\rangle \end{array}$$

where both $n, k \geq 0$ are arbitrary. We write x in place of $x\langle\rangle$—this captures $k = 0$.

The elements of the syntactic category of terms are also called "forests". The index *co* means that the grammar is read coinductively. There are two clauses that embed lists into the codata type. It therefore presents at least the challenges of non-wellfounded "rose trees" (finitely-branching unlabeled trees without a bound on the branching width). The scenario comes from [13, Section 3.2], and we plan to study it with our formalization. The typing rules for these expressions are given in Fig. 1, with Γ ranging over finite(!) typing contexts. They are the usual implication introduction and a vectorized implication elimination (down to atomic types p), and the obvious rule for typing alternatives – and all rules are read coinductively. A well-typed such term hence *locally* conforms to intuitionistic implicational logic. For illustration, we give a well-typed forest in graphical form (the much easier example of Church numerals is found in [12]). Let $\mathsf{THREE} :\equiv ((0 \to 0) \to 0) \to 0$. This is the simplest type of rank (i. e., nesting depth) 3. We define a closed forest of type THREE in Fig. 2 [13, Example 16]. $f@$ is short for $f\langle N\rangle$ with N given by where the arrow points to. The "decontraction operation" in the back link resides on the meta-level and is specific to the summation in this example grammar: $[(y+z)/y]$ (written $[y : 0, z : 0/y : 0]$ in the cited paper) is decontraction and says that every occurrence of y has to be replaced by a sum once with y and once with z in place of the original y. This forest representation can be seen as a formal approach to the informal concept of "inhabitation machines" [4, pp. 34–38]. All the inhabitants of THREE can be read off this forest: omitting types, they are of the form $\lambda f.f\langle\lambda y_1.f\langle\lambda y_2.f\langle\cdots\langle\lambda y_n.y_i\rangle\cdots\rangle\rangle\rangle$, with $1 \leq i \leq n$. The individual inhabitants are wellfounded, but the forests representing the entire search spaces (for all simple types) are obtained coinductively in [13].

We use a generic construction of syntax such as the forests of this scenario that is based on category theory.

3 The categorical framework for syntax with substitution

We construct non-wellfounded syntax from final coalgebras in suitable functor categories, hence based on category theory. We do not claim any originality on this construction. However, we benefit from that abstract view in order to construct a monadic substitution operation, i. e., a meta-level operation that is not specific to the application scenario presented above (in contrast to the decontraction operation it features). The qualifier "monadic" implies that the generic approach includes proving the monad laws. For the case of wellfounded syntax, this is well-established in the literature (see, e. g., [3]). For the non-wellfounded case but without types, this has also been done before [9]. What we are aiming at is to generalize the theory to the level of monoidal categories, where the monadic substitution operation appears as a monoid. On this level of abstraction, the representation of all terms as a whole is just by one object of the given monoidal category, thus abstracting away from context/scope and typing details. Concrete monoidal categories then allow for different forms of representation of syntax. For *wellfounded* syntax, there is a long line of research originating in [6]. For simply-typed wellfounded syntax see, e. g., Zsidó's PhD work [16]: in its Chapter 5, the monoidal category corresponding to the framework of [6] is laid out in detail for simple types, and its Chapter 7 compares it with the monoidal category of endofunctors over a slice category. The latter is close to the concrete instance we sketch in Section 4 of the monoidal approach to non-wellfounded syntax we describe here. To the best of our knowledge, *non*-wellfounded syntax with variable binding has never been studied before on this level of generality.

Our approach is to generalize the notion of heterogeneous substitution systems [11] from endofunctor categories to monoidal categories. Those systems (abbreviated HSS) were meant as a tool to construct monadic substitution both for wellfounded and non-wellfounded syntax – by a common abstraction that serves as a pivotal structure between initial algebras and final coalgebras, respectively, on the input side and the substitution monad as output. We call the generalization *monoidal heterogeneous substitution systems* (MHSS). For their definition, we need as background from category theory, besides monoidal categories, the actions of monoidal categories on categories, in particular the action of the (monoidal-)pointed objects of a monoidal category on its underlying category. The strength notion for actions in this case gives the concept of pointed tensorial strength. All these ingredients have been considered before for the sake of representation of wellfounded syntax [5, Section I.1.2][8, Section 5.2.1].

Our core contribution is to make the step to non-wellfounded syntax on the more abstract level of monoidal categories. As mentioned before, the aspects "non-wellfounded" and "abstract level of monoidal categories" were known separately. One might call our construction the "pushout" of the two techniques for those two aspects. Our technical development (cf. [12, Section 3]) demonstrates the pivotal role of MHSS: from a final coalgebra, a MHSS is constructed, and from a MHSS, a monoid is constructed, which abstracts away from monadic substitution.

The development of monoidal heterogeneous substitution systems is also discussed in an expository paper by Lamiaux and Ahrens [10]. That paper focuses exclusively on wellfounded syntax and discusses, in particular, the generalization of Hirschowitz and Maggesi's signatures [7] to the monoidal setting.

4 Non-wellfounded syntax for multi-sorted binding signatures

We apply the general results to the mundane endofunctor scenario – which is hardwired into the definitions in [11], but which we *choose* here as our main instance, already for convenient comparison.

1. We describe simply-typed syntax with variable binding (of finitely many sorted variables in each constructor argument) as a multi-sorted binding signature – this part is not original.

2. Given a multi-sorted binding signature, we construct a signature functor H (deviating from [1] for technical reasons).
3. We prove ω-continuity of H and construct the coinductive syntax as inverse of a final coalgebra.
4. We construct a "lax lineator" between actions expressing pointed tensorial strength of H.
5. We construct a MHSS for (H, θ) by applying our general construction [12, Theorem 4] of MHSS from final coalgebras.
6. We construct an (H, θ)-monoid by applying our general construction [12, Theorem 3] of a monoid from an MHSS.
7. Finally we interpret the obtained monoid as monad (hence as monadic substitution) by instantiating, in this whole chain of steps, the monoidal category to the endofunctors.

The overall sequencing of steps is not different from the construction of wellfounded syntax [1] from a multi-sorted binding signature – that also builds on the heterogeneous substitution systems of [11]. In what we do here for non-wellfounded syntax, the abstract level (monoidal categories instead of endofunctor categories) only shines through the proofs.

The non-wellfounded syntax appears as the carrier of a final coalgebra whose existence we guarantee by ω-continuity. However, this analysis differs from the ω-cocontinuity needed for wellfounded syntax in [1], and this made us also modify the definition of the signature functor (in particular, it is differently decomposed). These applications are absent from [11]: there, besides giving the signature functor for untyped λ-calculus (with and without explicit flattening), concrete non-wellfounded syntax is not constructed; now, we provide this construction. While the generalization of the categorical framework from endofunctor categories to monoidal categories is not instrumental for this application, we considered it important to meet the challenge of getting the non-wellfounded analogue of the tools described in [1]. Our more abstract framework not only makes the general theory conceptually much clearer (as seen in [12, Section 3] as compared to [11]) but it also opens the way to instantiations to the monoidal category considered in [6] and its ramifications, so that non-wellfounded syntax with monadic substitution can also be described in that setting.

5 Formalization

All of the definitions and results mentioned in this abstract (except for the motivational application scenario in Section 2) are formalized and computer-checked in UniMath [15], a library of univalent mathematics based on the computer proof assistant Coq [14]. The full version [12] has its definitions and results annotated by Coq identifiers for the corresponding definitions and results in our library. These identifiers are hyperlinks leading to an HTML version of the proof code, so that the formal entities are only a click away from the reader of the paper. The formalization is not the main message of this contribution; however, we subscribe to the idea of a co-development and co-presentation of the ideas and their computer formalization.

6 Our results in a nutshell

Categorical semantics for languages with binding can be achieved on the level of monoidal categories. This was seminal work [6] involving Marcelo Fiore who conducted numerous works in this spirit afterwards. In those works, non-wellfounded syntax is not addressed. A categorical semantics of non-wellfounded syntax with binding appeared in [11] involving the first author. This was set concretely in endofunctor categories. The new definitions and results of the present paper lift the approach of [11] to the abstraction level of

monoidal categories – to reach the same abstraction level as the anterior seminal work for wellfounded syntax. We even offer a full type-theoretic formalization of the results on the abstract level. By (non-trivial) instantiation, we get a tool chain from multi-sorted binding signatures to certified monadic substitution for non-wellfounded syntax and thus the non-wellfounded counterpart to the tool chain described in [1] involving two of the present authors. Such a tool chain is absent from [11] even on the informal level (neither multi-sorted binding signatures nor unsorted binding signatures are considered). Furthermore, our approach is now general enough so that the monoidal category underlying [6] and its ramifications can also be studied concerning non-wellfounded syntax and substitution for it. We refer to [12, Section 6] for more on related work.

References

[1] Benedikt Ahrens, Ralph Matthes & Anders Mörtberg (2022): *Implementing a category-theoretic framework for typed abstract syntax*. In Andrei Popescu & Steve Zdancewic, editors: *CPP '22: 11th ACM SIGPLAN International Conference on Certified Programs and Proofs, Philadelphia, PA, USA, January 17 - 18, 2022*, ACM, pp. 307–323, doi:10.1145/3497775.3503678.

[2] Guillaume Allais, Robert Atkey, James Chapman, Conor McBride & James McKinna (2021): *A type- and scope-safe universe of syntaxes with binding: their semantics and proofs*. *J. Funct. Program.* 31, p. e22, doi:10.1017/S0956796820000076.

[3] Thorsten Altenkirch & Bernhard Reus (1999): *Monadic Presentations of Lambda Terms Using Generalized Inductive Types*. In Jörg Flum & Mario Rodríguez-Artalejo, editors: *Computer Science Logic, 13th International Workshop, CSL '99, Proceedings*, *Lecture Notes in Computer Science* 1683, Springer, pp. 453–468, doi:10.1007/3-540-48168-0_32.

[4] Henk Barendregt, Wil Dekkers & Richard Statman (2013): *Lambda Calculus with Types*. Perspectives in Logic, Cambridge University Press.

[5] Marcelo P. Fiore (2008): *Second-Order and Dependently-Sorted Abstract Syntax*. In: *Proceedings of the Twenty-Third Annual IEEE Symposium on Logic in Computer Science, LICS 2008, 24-27 June 2008, Pittsburgh, PA, USA*, IEEE Computer Society, pp. 57–68, doi:10.1109/LICS.2008.38.

[6] Marcelo P. Fiore, Gordon D. Plotkin & Daniele Turi (1999): *Abstract Syntax and Variable Binding*. In: *14th Annual IEEE Symposium on Logic in Computer Science, Trento, Italy, July 2-5, 1999*, IEEE Computer Society, pp. 193–202, doi:10.1109/LICS.1999.782615.

[7] André Hirschowitz & Marco Maggesi (2010): *Modules over monads and initial semantics*. *Inf. Comput.* 208(5), pp. 545–564, doi:10.1016/J.IC.2009.07.003. Available at `https://doi.org/10.1016/j.ic.2009.07.003`.

[8] Chung-Kil Hur (2010): *Categorical equational systems : algebraic models and equational reasoning*. Ph.D. thesis, University of Cambridge, UK. Available at `http://ethos.bl.uk/OrderDetails.do?uin=uk.bl.ethos.608664`.

[9] Alexander Kurz, Daniela Petrisan, Paula Severi & Fer-Jan de Vries (2013): *Nominal Coalgebraic Data Types with Applications to Lambda Calculus*. *Log. Methods Comput. Sci.* 9(4), doi:10.2168/LMCS-9(4:20)2013.

[10] Thomas Lamiaux & Benedikt Ahrens (2024): *An Introduction to Different Approaches to Initial Semantics*. arXiv:2401.09366.

[11] Ralph Matthes & Tarmo Uustalu (2004): *Substitution in non-wellfounded syntax with variable binding*. *Theoretical Computer Science* 327(1-2), pp. 155–174, doi:10.1016/j.tcs.2004.07.025.

[12] Ralph Matthes, Kobe Wullaert & Benedikt Ahrens (2023): *Substitution for Non-Wellfounded Syntax with Binders through Monoidal Categories*. *CoRR* abs/2308.05485. arXiv:2308.05485.

[13] José Espírito Santo, Ralph Matthes & Luís Pinto (2021): *A coinductive approach to proof search through typed lambda-calculi*. *Ann. Pure Appl. Log.* 172(10), p. 103026, doi:10.1016/j.apal.2021.103026.

[14] The Coq Development Team (2023): *The Coq Proof Assistant, version 8.17*. Available at `https://zenodo.org/records/8161141`.

[15] Vladimir Voevodsky, Benedikt Ahrens, Daniel Grayson et al. (2021): *UniMath — a computer-checked library of univalent mathematics*. Available at `http://unimath.github.io/UniMath/` .

[16] Julianna Zsidó (2010): *Typed Abstract Syntax*. Ph.D. thesis, University of Nice, France. `http://tel.archives-ouvertes.fr/tel-00535944/`.