    Dear all,

    apparently, all three reviewers agree that they *"found the paper somewhat difficult to read and often times confusing."* I am sorry, indeed I wrote it in haste, for the fear of forgetting this task. Remark that Referee 3 also said that **" I found the results and development of such beautiful λ-calculus formalism for sure of interest for the TCS community"**. Yet, he just gave score 1, for the reasons explained above. I still consider the paper promising, but we have to re-write it. For instance, we could aim at **CSL 2026**:

* ***Abstract submission:*** *15 July 2025*
* ***Paper submission:*** *21 July 2025*
* ***Notification:*** *20 Ocotber 2025*
* ***Final version:*** *30 November 2025*
* ***Conference (Paris):*** *23-28 February 2026*

    In this case we would have two months of time. But I think we should **not** hurry this time. We should all read and improve the paper first, and if we feel the paper is not ready yet, then we can aim to the next congress. I can implement all referees suggestions first, then we all can start re-writing it.

    What do you think? Stefano

Il 16/05/2025 11:46, MFPS 2025 ha scritto:

Dear Stefano Berardi,

we regret to inform you that your paper

6 A Cyclic Simply Typed Lambda-Calculus with Fixed Points

has been rejected for MFPS 2025.

We have received 28 submissions of which 18 have been accepted for publication.

The Program Committee has made an effort to provide useful feedback to the authors. However, note that this feedback does not always fully capture the detailed discussions that took place beyond the written reviews.

We thank you for choosing to submit your work to MFPS 2025. We hope that you will find the reviews useful and that you will attend the conference regardless of the disappointing outcome of the review process for your submission.

Best regards,

Clemens and Stefan.

SUBMISSION: 6

TITLE: A Cyclic Simply Typed Lambda-Calculus with Fixed Points

----------------------- REVIEW 1 ---------------------

SUBMISSION: 6

TITLE: A Cyclic Simply Typed Lambda-Calculus with Fixed Points

AUTHORS: Stefano Berardi, Ugo de' Liguoro, Daisuke Kimura, Koji Nakazawa and Makoto Tatsuta

----------- Overall evaluation -----------

SCORE: -1 (weak reject)

----- TEXT:

This paper deals with a circular version of lambda-calculus, with self-referential terms typed via circular proofs. The global trace condition, which is a particular case of conditions from previous works, guarantees validity of such proofs by ensuring that following an infinite path yields an infinitely decreasing sequence of integers, so in practice all computations are finite.

The main results are :

-closed terms of type N reduce to actual numerals

-a form of strong normalization when reductions avoid certain subterms (the ones using predecessor)

-the existence of some limit object for reductions in general.

The paper is not well-written, and it seems clear that it was not carefully proofread. As detailed below, besides the many typos, the explanations are often convoluted and the writing style makes the paper hard to read. In many places the paper goes into excruciating unnecessary details, while some important notions (e.g. the limit) are not defined, and some important results (e.g. subject reduction) are not proved.

The existence of a limit normal form is actually not proved but just stated and shown on an example.

The introduction gives very little context and dives right into definitions.

There are only 6 cited papers. Similar lines of research should be cited, and the present work should be situated in the context of other works on circular lambda calculus and circular proofs. For instance after the following papers on circular lambda-calculi seem relevant:

\*Cyclic lambda calculi, by Ariola and Blom

\*Lambda Calculus with Explicit Recursion, by Ariola and Klop

\*Expressibility in the Lambda Calculus with Letrec, by Grabmayer and Rochel

\*Unfolding Semantics of the Untyped λ-Calculus with letrec, by Rochel

Despite the relevance of the topic and the potential interest of the results, I do not feel the paper in its present form is ready for publication.

Below is a list of typos and detailed remarks.

Abstract:

-syntax\*es\* (plural missing in several places)

Introduction:

-will correspond to \*a\* subset

-strong\*ly\* normalizes

-"to all a λ- calculus"

FIN QUI 27/05/25

Section 2:

-The definition of binary trees in unnecessarily long and and convoluted. Several unusual notations make the reading more difficult. Ant the useful and a bit less trivial notions like path are not defined at all.

-notation l is used for lists in {1,2}\* and for labels in L

-def 2.2 "with A,B also A->B" awkward

-def 2.4 use \mid instead of vertical bar |, or add spaces, to avoid "t⌈l|l"

-finitely many [pairwise] distinct variables (no need for "pairwise")

-judg[e]ments (e should be removed)

-"parwise" (should be removed anyway)

-"is the pair of" awkward

-def 2.5

item 5: recall what are the x\_i

item 6: context\*s\*. Also why not write simply FV(Γ)⊂FV(Γ') ?

item 7: why say for all x\_i^{A\_i}=x^T when there can be only one, as per item 1. ? It would be simpler to say that x^T:T is removed from the list if present.

item 8: you redefine inclusion from item 6, so no need for two items

"As usual, we informally write a sequent (Γ, A) for t in the form Γ ⊢ t : A.": this notation was already introduced in item 4 of def 2

-informal definition of t[u/x]: why bother defining the notation T⌈l if you won't use it for term substitutions, the only place where it would be kind of useful. All this formalism to end up having informal definitions when it starts to matter seems counter-productive.

-"according if" -> according to whether

-but we we

-"The information included in ap\_v is easily lost" -> not clear what you mean

-Def 2.6:

you are not following your definitions of sequents and typing judgments: rules are defined with typing judgments, not sequents (same issue later)

item 4: the way it is written, you need x^A in Γ in the premise, which seems inconsistent with item 3.

item 8: we still don't know what the semantic of cond(f,g) is supposed to be, only a vague indication that it is a "test on zero"

-could \*be\* derived

-when whenever

-are no\*t\* instances

-does ",say," mean "that is to say" here ? this sentence is not clear: why do you isolate these two rules specifically ? what you call "leftmost" is actually not clear at all, it should be explicited.

-the notation Π : Γ ⊢ t : A is unfortunate, as the same symbol ":" occurs at very different levels.

-Γ ⊃~ FV(t) does not typecheck with your definitions.

-"Proofs not using the weak-rule are unique" -> explicit what you mean: you fix a term t (and a type A ? a context Γ ?) and look at proofs of Γ ⊢ t : A

-typesA

-a set of well-formed term\*s\*

-def 2.11,

item 8: "depth" should be cond-depth

item 10: "safe-normal" loses the hyphen just after

-"right- hand-side" -> right-hand side

-"β- redexes" -> β-redexes

-will-express

-the the

-until possible -> while possible

Section 3:

-infinitelydecreasing

-can exist[s]

-the arguent\*s\* of its premise

-the description of connections between arguments is hard to read and unclear. One of the problems, besides general hardness to parse, is that the scope of definitions of objects is not obvious at all (e.g. u in item 2 is referring to the u defined by item 1, but t is changed to f).

-a a free

-By definition \*of\* unfolding

-the definition of connections takes up two pages of cumbersome formalism, it could be compressed and explained more efficiently. Keeping track of explicit indexes and doing the computations does not help the reader figure out what is encoded. Just giving a picture for each rule, showing the connections, would be clearer and would save a lot of space and effort.

-if [t=]λx.v is obtained by a rule

-x3 distinct from x1,x3 (you mean x2 ?)

Section 4:

-"Assume i = m,...,n." -> Assume i \in {m,...,n}

-"We believe that the global trace condition is decidable quite fast for realistic λ-terms." -> better say nothing that something so vague

-A first example of term of CT-λ. -> not a sentence

-the variable y appears in the footnote 1, it should be z (or more logically change z to y everywhere)

-Example 4.2: I don't see what the equations up to ~ bring, you could just give directly the term for the iterator "it", it is even clearer than the sum case that you gave directly in the previous example. It would be more interesting to comment on whether it can be alternatively defined the other way by adding the last f inside instead of outside, by It=λf,a.cond(a,λx.It(f,f(a))(x))

-Example 4.3: there are too many definitions of i: i is first defined as i=λx.cond(b, λy.i), but then you add i=c( it(x), i(S(x))(y) ), it does not seem to make sense. I suspect you just reused the name i instead of a different name, which is very unfortunate since the reader does not know which is which. Moreover, the definition λx.cond(b, λy.i) does not match the first unfolding in the proof, which is cond(b, i) instead of cond(b,λy.i). At this point I'm giving up trying to fix the example (which I'm convinced can be done, but this should not be up to the reader)

-how difficult \*it\* is

-The nested substitution produce\*s\*

-Better Code [it] is Easier to Analyse

-trivial variant\*s\*

-can compute the \*same\* function

-why is the last m' bold in ack1

-way we add

Section 5:

-the results of this section are not proved anywhere, there is no proof or sketch or anything in the body nor in the appendix. If they are so easy that none of that is necessary, it seems weird to call them Lemma, Theorem, Proposition.

-so is Π′-> so does Π′

Section 6

-Def 6.1: dot at end, capitalize first letter of next sentence "To put otherwise"

-For, we

-for all total[s]

-cannot exist[s]

-Prop 6.6: make clear that items 1 and 2 are conclusions, e.g. by adding "Then, " before

-by reductions, by application -> choose singular or plural

-Corollary 6.10: point 2 directly implies point 1, and point 3 seems a direct reformulation of point 2, so points 1 and 3 can be removed.

-what do you call a "limit" ? it should be defined.

-"finiteness of a notation": awkward, it is not the "notation" that is finite, notation is not a well-defined mathematical notion.

References:

-some authors are just initials

----------- Reviewer's confidence -----------

SCORE: 4 ((high))

----------------------- REVIEW 2 ---------------------

SUBMISSION: 6

TITLE: A Cyclic Simply Typed Lambda-Calculus with Fixed Points

AUTHORS: Stefano Berardi, Ugo de' Liguoro, Daisuke Kimura, Koji Nakazawa and Makoto Tatsuta

----------- Overall evaluation -----------

SCORE: 0 (borderline paper)

----- TEXT:

This paper presents a cyclic simply typed lambda-calculus for Godel's system T, based on existing circular syntax for that system. The system allows for the representation of infinite terms, by introducing recursive definitions that can unfold infinitely many times. Some adequacy and normalisation properties are proved.

Within the system, the authors consider a subset (CT-lambda) corresponding to finite well-typed terms (which the authors claim to correspond to the usual Godel's system T terms). A second subset is considered, which satisfies a global trace condition (following a notion by Brotherston adn Simpson and adapted to System T by Das, Kuperberg, Pinault and Pous), and which is intend to give an alternative semantics to CT-lambda.

While I am familiar with Godel's system T, this was my first contact with a circular syntax for the system, and while I do find this approach to be interesting, I find that there are some issues with this paper. First, there is no proper comparison with the existing circular versions of T (the authors only mention that they are using abstractions instead of combinators, but there is no discussion on the advantages/disadvantages of this approach). And although I can see the relevance in using a syntax more familiar to type theorists (this seems to be the motivation), I still think that a proper discussion on related work is necessary.

The paper also lacks in terms of format: there are several inconsistencies in notation (see comments below); the introduction is very short, failing to give a proper overview of the paper and there is no conclusion (the paper ends abruptly with a proposition). The authors fail to properly introduce notations that are used, leaving the reader to extract their meaning from the context, but on the other hand there are sentences that are repeated with no need (for example, the notions of one step, zero or more steps; and one or more steps (of reduction), are mention 3 times in the paper, when 1 would be enough).

Regarding the future work, there is a brief discussion at the end of the introduction, where the authors mention something that I think should have been explored in this paper: which is the relation between closed CT-lambda terms and computable functions in Godel's T.

In conclusion, while I find the idea of the paper interesting and appropriate for MFPS, I believe the paper still needs some work.

SMALL TYPOS AND SPECIFIC COMMENTS

- page 1: You claim that type variables are only used to provide examples of infinite terms and play a minor role in the paper. Could they then be eliminated without compromising the semantic model? If so, please clarify that.

- page 1/2: you seem to be using different notations when referring to Godel's System T. Please be consistent.

- page 2: "... reductions not in the right-hand side of any cond (conditional) strongly normalize..."

- page 2: your definition for binary trees would strongly benefit from an example.

- page 2: what does it mean for a list l to be less or equal than a list m?

- page 2: "l is (a) maximal element"

- page 3: you seem to be using different notations when referring to set of terms Lambda. Please be consistent.

- page 3: "An example of (a) regular term (that) is..."

- page 4: I agree that there is redundancy in your notation for contexts. I understand that you are following a Church presentation for the type system, but couldn't you just assume that a typing context is a subset of var (defined in Def 2.2.)? You claim that this choice is to be consistent with the notation used in type theory, but there is little in common in the way you latter present the type system and the way that is usually used by type theorists.

- page 4: in Def. 2.5 (point 6), to be consistent you need to add the superscripts A'\_1,...,A'\_n to x\_1,...,x\_n. In point 7, since a context is a list of pairwise distinct variables, why do you need the forall condition when removing a typing assignment?

- page 5: in Def. 2.6, why do you need weakening? Isn't weakening implicit in your var rule? You latter claim that this is a choice to regularity. Could you then have var rule simply as x^A:A |- x^A:A?

- page 5: "could (be) derived"

- page 5: "associated to the children node(s)"

- page 6: be consistent with the notation for almost-left-finite (either using capitals or not).

- page 6: isn't proposition 2.10 just a consequence of having a Church-style system? That is, doesn't it hold even without the almost-left-finite condition?

- page 6: you use a,b,f,g,t,u to denote terms. Is there some logic to the use of such range of letters? If so, explain it. Also in point 8 of this definition, do you mean cond-depth?

- page 6: in example 2.11 there is an extra "." between cond and its parameters that was not introduced in the syntax. Also, please remove the extra space in "right- hand-side" and "beta- redexes".

- page 7: in example 2.3 you seem to be using natural numbers as an abuse of notation for terms of the form S^n 0. Please clarify that in the text.

- page 7: "infinitely(-)decreasing... then no infinite computation can exist".

- page 7: I do not understand what you mean with A=A\_1,...,A\_n when abbreviating A\_1,...,A\_n -> B with A -> B. According to your definition of types I believe who mean that you abbreviate A\_1 -> ... -> A\_n -> B with A -> B. Also I assume you mean Gamma = {x\_1^A\_1: A\_1,\dots x\_n^A\_n: A\_n}. If you want to use this abuse of notation, please introduce it properly.

- page 8: although you are explaining why you use outlined N for the type N to highlight a connection, I would not use the same notation that you use for the set of natural numbers. Maybe use \mathbb{N} for that, as is usual in literature.

- page 8: in example 3.1 I believe you mean "x\_3 is a variable different from x\_1,x\_2".

- page 9: be consistent when referring to "global trace condition" (either using capitals or not).

- page 9: you say that you believe that global trace condition is decidable quite fast for realistic lambda-terms. What do you mean by fast? What do you mean by realistic? What sustains this claim?

- page 9: you seem to be using GTC to refer to both the set of terms that satisfy the global trace condition and to the condition itself. This is confusing.

- page 9: "A first example of (a) term of CT-lambda".

- page 9: finally in example 4.1 the need for the weakening rule is made clear. Why not present this example earlier in the paper and refer to it here?

- page 10: "for all numeral(s) n"

- page 10: you are skipping the superscripts for the types in these examples. I would suggest that you state earlier in the paper, that you will omit those superscripts whenever they are clear from the context. I believe this would be helpful in other places where you made the same omission.

- page 10: in example 4.3, I cannot parse the sentence "We solve the recursive equation for I...". And again, you are very loose with your syntax: \f,a.i does not correspond to any term in you syntax, as well as c(a,l). As I understand, you are using tuples with terms that are actually "curried". Please be more clear about this.

- page 11: what does it mean "Thus, the total time is infinite, a limit process".

- page 11: "the normal form we obtain in this way (is) not normal".

- page 12: please correct "way we add more connections" and "total automatically".

- page 12: "if Pi satisfies the global trace condition, so (does) Pi'".

- page 12: when you say n in N, I assume you are referring to the term S^n 0 that is typed with type N. Because you use sometimes actual natural numbers in examples, you should perhaps clarify this earlier in the paper.

- page 12: there is a dot missing at the end of Def. 6.1, and the next paragraph should start with a capital letter.

- page 13: correct "For, we define..." and "relation predecessor relation"

- page 15: again, there is some confusion between the term S^n in N and n in (outlined) N (the set of natural numbers). I believe the notion of n-safeness is defined for the latter not the former.

----------- Reviewer's confidence -----------

SCORE: 3 ((medium))

----------------------- REVIEW 3 ---------------------

SUBMISSION: 6

TITLE: A Cyclic Simply Typed Lambda-Calculus with Fixed Points

AUTHORS: Stefano Berardi, Ugo de' Liguoro, Daisuke Kimura, Koji Nakazawa and Makoto Tatsuta

----------- Overall evaluation -----------

SCORE: 1 (weak accept)

----- TEXT:

# Summary

The paper develops a simply typed λ-calculus style of syntax in which terms

syntax is allowed to be cyclic and therefore be faithfully represented by

infinite trees.

It turns out that such a development is no trivial endeavor and the paper tackle

this by providing a full treatise on how to formally define and reason about

such structure.

Central to the notions of terms and types lies also the notion of reduction,

which essentially applies beta reductions ``safely`` (since terms can be infinite).

A normalization result for the class of safe terms is presented.

This is expected since the class of 0-safety terms are exactly those well-behaved ones.

The paper proceeds to develop the search for those classes of infinite terms

which are well-behaved, and introduces the notion of global trace.

Important results are proved for such notions.

# Assessment

While I found the results and development of such beautiful λ-calculus formalism

for sure of interest for the TCS community, I found the paper somewhat difficult

to read and often times confusing.

I believe major work needs to be done on the writing and presentation of the paper.

Mainly:

- the introduction is a bit unorganized, in my opinion, i.e., why start it by

defining simple types (which is common enough)? One is usually expecting

some intuitions on the work that will be done;

- connections to related work seems to undersell the actual work on this paper,

initially I thought it would only add lambdas to Gödel's system T.

However, the paper develops way more than "just adding lambdas";

- the order to which information is presented is also a bit confusing to me,

it is common that future tense is used to discuss what will be done further down

the line (without directions to it) or an explicit reason to such statements;

- definitions are quite long, sometimes with 10 items. I believe a lot of what

is defined on such enumerates can be stated in the text since some of them

are common enough or not important to the main definition being treated;

- while there are some definitions with a high number of items defining

"common" structures for those versed in λ-caluclus and type theory,

some that are not so common lack explanations and intuitions

(e.g. Def. 2.1, discussed below);

- finally these flaws make the paper hard to read in "one go" and really

appreciate its results.

## Detailed comments

The authors can find some comments I found relevant to point out in details below.

Definition 2.1 can be confusing to someone not used to this representation.

Suggestion for improvements:

Item 1.

- Make it clear that the ordering l ≤ m on item 1 is the prefix order,

so a notation like ≤\_{prefix} or state that explicitly (it is mentioned first only in item 3)

- this makes the trees posets under the prefix ordering

- add a sentence or two explaining your choice of tree data structure

\* that is, trees are represented the set of paths that can be taken from

the root (in which case nil represents the empty path)

\* that's why it should be closed under prefix

\* but the most important of all: you want to represent infinite terms,

and this (seems to me) that is a good way to represent them rather to use

the classical inductive definition

- is this definition new on this paper? Inspired by some other definition?

Please add reference if that's the case.

Item 2.

- Please fix notation and the ∈ on this item.

Most importantly, what is l⋆(i) ?

Item 4.

- Notations are introduced without any explanation of what they mean.

What's l⋆m ?

On item 2 there is an l⋆(i) which might make this one confusing too,

if not properly explained.

- What happens to nil⋆nil?

I can get intuitively what it is meant by subtree here

- (take all the sub-paths starting from a node l closing them by prefix),

that will give the subtree (and the root is now relative) to the "position"

of the initial node in the bigger tree.

Item 5. What is meant by l = (i)? You mean the list l = [1] or l = [2]?

Also, no notation for list was introduced up to now.

Is it, [ l₁ ; ... ; lₙ ] ? That would be my guess, but then I don't know what is

meant by l = (i).

Lastly, adding examples of finite trees would be nice

and provide better intuition for the rest of the paper.

Definition 2.5.

I don't understand the need for annotating variables directly with their types

and putting terms under a context at the same time.

It is indeed redundant, as stated in the paper.

If you define terms separately from their types and don't annotate variables,

context the context give the type information for the variables and together

with the typing information of each constructor one can build the type derivation

of the term, so essentially writing a term-in-context.

The paper states:

"our variables already have a type superscript,

so in fact a type assignment x^T : T is a redundant information for our variables

(not for our terms)."

Why though? It is very well possible to design the judgement typing rules

for a type statement of the form t : A, without a context Γ.

Often called "λ-terms a la Church".

When no explicit anotation is used, it is often called the curry version.

(See for instance LAMBDA CALCULUS WITH TYPES by Barendregt, Dekkers, and Stateman)

Mixing both, in my opinion, makes the development redundant and the notation heavier.

If there is any other reason to keep both styles, please make it clear already

in this section.

Notice, however, that both versions has differences when semantics is given:

for implicit typed terms Γ ⊢ t : A, one need to also define a semantical interpretation

for contexts.

Definition 2.6: Why not present the typing rules in natural deduction form?

That is, for instance:

x : A ∈ Γ

--------------

Γ ⊢ x : A

Definition 2.11 is quite long and could be presented more nicely.

On Section 3 traces are introduced, it would be important to give

examples to motivate this. I believe the paper would benefits a lot from

this since it is not so easy to grasp why are you taking such traces at a first

glance.

In the examples of this section I don't understand the use of such colors.

Section 4.

This section got me curious, since you have type variables in the type syntax:

can a form of parametric polymorphism be added here directly?

It seems to be that that would be the case easily,

so why not give terms like map (extending the syntax a bit to allow lists)

or even foldl (foldr seems to be more difficult, but i am not so sure).

Why not a Conclusion/Future work section?

----------- Reviewer's confidence -----------

SCORE: 4 ((high))