CT-λ, a cyclic simply typed λ-calculus with fixed points

Stefano Berardi

Abstract

We explore the possibility of having a circular syntax directly for λ-abstraction, instead of interpreting λ-abstraction through combinators. The use of binders instead of combinators for defining terms could be more familiar for researchers in the field of Type Thepry

1 Introduction

We will introduce $\mathbb{A}$, a set of infinite terms whose types are: the atomic type N, possibly type variables, and all types $A \to B$ for any types A, B. The terms of $\mathbb{A}$ are possibly infinite trees representing expressions defined with 0,$\mathtt{S}$,$\mathtt{ap}$, variables, the binder λ for defining maps, the binder $\mathtt{cond}$ for arithmetic conditional. If we have no type variables, the trees in $\mathbb{A}$ represent partial functionals on N.

Then will consider a subset $\mathtt{CT}$-λ of $\mathbb{A}$ which we call the circular version of Gödel system $\mathbb{T}$. Differently from all previous circular version of $\mathbb{T}$, our system uses binder instead of combinators.

$\mathtt{CT}$-λ consists of all trees in $\mathbb{A}$ which are well-typed, satisfy a condition called global trace condition, and are regular (they only have finitely many sub-trees). We will prove that the trees in $\mathtt{CT}$-λ whose only atomic type is N represent exactly the total computable functionals on N definable in Gödel system $\mathbb{T}$ using N as only atomic type.

2 The set of infinite λ-terms

We define the set $\mathbb{A}$ of infinite terms, well-typed terms and a notion of reduction for them.

Definition 2.1 (Types and infinite terms in $\mathbb{A}$)

1. *The types of $\mathbb{A}$ are all types we can define with N and $\to$.*
2. *The terms of $\mathbb{A}$ are all possibly infinite trees we can define with x (variable name), $\lambda x : T.t$ (binder, abstraction), $\mathtt{ap}(t, u)$, 0, $\mathtt{S}(t)$ (successor of t), $\mathtt{cond}x.(f, g)$ (binder, arithmetical conditional).*

When $t = \mathtt{S}^n(0)$ for some natural number n we say that t is a numeral. We write $\mathbb{N}$ for the set of numeral.

$\mathbb{N}$ is yet another representation of the set of natural numbers. All numeral are finite terms of Λ. All finite simply typed λ-terms we can define with the rules above are finite terms of $\mathbb{A}$. The term $t = \mathtt{cond}x.(0, t)$ is in $\mathbb{A}$ and it is infinite. Its unique infinite branch is $t, t, t, \ldots$.

We define context and inherited context.

Definition 2.2 (Contexts of $\mathbb{A}$) *A context is any finite list $\Gamma = x_1 : A_1, \ldots, x_n : A_n$ of pairwise distinct variables and types. Given any context Γ, any $t \in \Lambda$, any subterm u of t we define an inherited context for u, of the form Γ, Δ for some Δ.*

1. *t inherited the context Γ.*
2. *Any binder on x subtracts the variable x from the context: if $t = \lambda x : T.u, \mathtt{cond}x.(f, g)$ has context Γ, Δ, then u and g have context $\Gamma, \Delta, x : T$.*
3. *In any other case the context of a term and an immediate subterm are the same: If $\mathtt{cond}x.(f, g)$ has context Γ, Δ, then f has context Γ, Δ.*

We define the statement: *"t has context Γ and type A"*, written $\Gamma \vdash t : A$, and of proof Π of $\Gamma \vdash t : A$.

The typing rules are the usual ones but for the conditional binder `cond`, and for a fresh rule η-`rule`. η-`rule` corresponds to an η-expansion but introducing a global variable name.

We have a single structural rule merging Weakening, permutation and variable renaming.

Definition 2.3 (Well-typed terms of $\mathbb{A}$) *Assume $\Gamma = x_1 : A_1, \ldots, x_n : A_n$, $\Delta = y_1 : B_1, \ldots, y_n : B_m$ are sequents of length n, m respectively. Suppose $f : \{1, \ldots, n\} \to \{1, \ldots, m\}$ is any injection, compatible with types in Γ, Δ: $A_i = B_{f(i)}$ for all $1 \leq i \leq n$.*

1. *Structural rule* `struct`(f). *If $t : \Gamma \vdash T$ then $t[y_{f(1)}/x_1, \ldots, y_{f(n)}/x_n] : \Delta \vdash T$*
2. η-`rule`. *If $x \notin \Gamma$ and $f : \Gamma \vdash N \to T$ then $f(x) : \Gamma, x : N \vdash T$.*
3. *Var-rule. If $x : A \in \Gamma$ then $x : \Gamma \vdash A$.*
4. `ap`*-rule. If $\Gamma \vdash f : A \to B$ and $a : \Gamma \vdash A$ then $f(a) : \Gamma \vdash B$.*
5. *λ-rule. If $\Gamma, x : A \vdash b : B$ then $\lambda x : A.b : \Gamma \vdash A \to B$.*
6. *0-rule.* $0 : \Gamma \vdash N$
7. `S`*-rule. If $t : \Gamma \vdash N$ then* $\mathtt{S}(t) : \Gamma \vdash N$.
8. `cond`*-rule.* `cond`. *If $f : \Gamma, x : T \vdash T$, $x \notin \mathrm{FV}(f)$ and $g : \Gamma, x : T \vdash T$ then* $\mathtt{cond}x.(f, g) : \Gamma \vdash N \to T$.

We write $\Pi : \Gamma \vdash t : A$, and we say that Π is a proof of $\Gamma \vdash t : A$ if Π is an assignment of one sequent $\Delta \vdash B$ to each subterm u of t, in such a way that u is the consequence of some rule and $\Gamma \vdash A$ is assigned to t.

1. *$\Gamma \vdash t : A$ is true if and only if it has some proof.*
2. *$t \in \mathbb{A}$ is well-typed if it has some proof.*
3. `Wtyped` *is the set of well-typed $t \in \mathbb{A}$.*

We write $\vdash t : A$ for $\emptyset \vdash t : A$.

η-`rule` is a particular case of `ap`. When we substitute $x : N$ with some $\Gamma \vdash a : N$ then we replace η-`rule` with `ap`-rule.

An example: if $t = \mathtt{cond}x.(0, t)$, then $\vdash t : N \to N$.

Our goal is to provide a set of well-formed term for $\mathbb{A}$ and interpret them as partial functionals, and total functionals if we add more conditions. Our first step is to provide reduction rules for $\mathbb{A}$.

Definition 2.4 (reduction rules for $\mathbb{A}$)

1. *β: $(\lambda x : A.b)(a) \mapsto_\beta b[a/x]$*
2. $\mathtt{cond}x.(f, g)(0) \mapsto_{\mathtt{cond}} f$ *and* $\mathtt{cond}x.(f, g)(\mathtt{S}(t)) \mapsto_{\mathtt{cond}} g[t/x]$.
3. *$\mapsto$ is the context and transitive closure of $\mapsto_\beta$ and $\mapsto_{\mathtt{cond}}$*
4. *We say that $t \mapsto_{\mathtt{safe}} u$, or that t reduces safely to u, if we never reduce in proper subterms of any* $\mathtt{cond}x.(f, g)$. *We call unsafe a reduction inside any* $\mathtt{cond}x.(f, g)$.
5. *A term is safe-normal if all its redexes (if any) are inside some* $\mathtt{cond}x.(f, g)$.

As example. If $u = \mathtt{cond}x.(0, (\lambda x.u)(n_0))$ is as above, then u is safe-normal, because all redexes in u are of the form $(\lambda x.u)(n_0)$ and inside a `cond`. However, there are infinitely many β-redexes inside u.

The reason for forbiding reductions inside $\mathtt{cond}x.(f, g)$ is that thorugh `cond` we will-express fixed-point equations. Therefore reductions for `cond` are in fact unfolding of the definition of a fixed point. As unfolding of a fixed point, they can can easily loop, and they are "unsafe". For this reason, we allow the minimum possible reductions concerning $\mathtt{cond}x.(f, g)$, those of the form $\mathtt{cond}x.(f, g)(0) \mapsto_{\mathtt{cond}} f$ and $\mathtt{cond}x.(f, g)(\mathtt{S}(t)) \mapsto_{\mathtt{cond}} g[t/x]$ for maximal `cond`-expression, and no reduction inside the arguments f, g of `cond`.

An example. If n is any numeral and $u = \mathtt{cond}x.(0, (\lambda x.u)(z))$, then $u(n) : N$ has infinitely many β-redexes inside `cond`, therefore infinitely many unsafe reductions are possible. There are only finitely many safe reduction from $u(n)$ instead: $u(n)$ reduces to $\lambda x.u)(z)(n-1)$, then to $u(n-1)$, then $u(n-1)$ reduces to $u(n-2)$ and so forth. After $n + n$ reductions we get $u(0)$. With one last reduction we get 0 and we stop.

3 The trace of the λ-terms

We define a notion of trace for possibly infinite λ-terms, describing how an input of type N is used when computing an output. The first step toward a trace is defining a correspondence between atoms in the proof that t is well-typed. We need first the notion of list of types and atomic types for a term.

Definition 3.1 (Integer maps) *1. The type list of t is $\vec{C} = \vec{A}, \vec{B}$.*

2. An atom index of t is any $j \in \{1, \ldots, n+m\}$ such that $C_j = N$.

3. We define $\mathtt{ins}(a, x) = x + 1$ if $x \geq a + 1$, and $\mathtt{ins}(a, x) = x$ if $x \geq a$.

We now define the atom correspondence in CT-λ.

Definition 3.2 (Atom correspondence in CT-λ) *Assume $\vec{A} = A_1, \ldots, A_n$, $\vec{B} = B_1, \ldots, B_m$ and $t : \vec{x} : \vec{A} \vdash \vec{B} \to N$ is a pseudo-term.*

1. *The type list of t is $\vec{C} = \vec{A}, \vec{B}$. An atom index of t is any $j \in \{1, \ldots, n+m\}$ such that $C_j = N$.*
2. *For each atom index k in t, each immediate subterms t' of t each atom index k' in t' we define the relation: "k', t' the successor of k, t". We require:*
   (1) if t is obtained by a rule $\mathtt{struct}(f)$ then $k = f(k')$ if $k' \leq n$ and $k = k' - n + m$ if $k \geq n + 1$.
   (2) $k' = \mathtt{ins}(n, k)$ if $t = f(a)$, $a : N$ and $t' = f$.
   (3) $k = k'$ in all other cases.

Definition 3.3 (Trace for well-typed terms in $\mathbb{A}$) *1. A path π in t is any list $t_0, \ldots, t_n$ of subterms of t such that $t_0 = t$ and each t_{i+1} is an immediate subterm of t_i.*

2. Assume $\pi = t_0, \ldots, t_n$ is a branch of t. A trace of π consists of an integer list $k_m, \ldots, k_n$ such that for each $i = m, \ldots, n$: k_i is an atom index of t_i, and if $i + 1 \leq n$ then k_{i+1}, t_{i+1} is the successor of k_i, t_i.

4 The circular system CT-λ

We define a subset CT-λ of the set Wtyped of well-typed term, by adding the global trace condition and regularity. For the terms of CT-λ we will prove strong normalization, church-rosser for terms of type N, and the fact that every term of type N is a numeral. As a consequence, terms CT-λ will be interpreted as total functionals.

From the atom correspondence we define the global trace condition and terms CT-λ.

We say that a tree is regular if it has finitely many subtrees. $t = \mathtt{cond}x.(0, t)$ is an infinite regular tree, with subtrees: t, 0.

Definition 4.1 (Global trace condition and terms of CT-λ) *1. Assume $\tau = k_m, \ldots, k_n$ is a trace of $\pi = t_0, \ldots, t_n$ and $i = m, \ldots, n$. τ is progressing in i if $t_i = \mathtt{cond}x.(f, g)$ for some f, g, and k_i is the index of the first argument the cond-rule, otherwise τ is not progressing in i.*

2. t satisfies the global trace condition if for all infinite paths π in t there is some infinitely progressing path τ in π.

3. Terms of CT-λ are all well-typed terms which are regular trees (having finitely many subtrees), and satisfy the global trace condition.

5 Examples of terms of CT-λ

5.1 The sum map

As a first example of term of CT-λ we can provide a term Sum computing the sum on N.

....
(Here we should insert the content of the board checking that Sum $\in$ CT-λ)
....

5.2 The Iterator

A second example. We define a term $\mathtt{It}$ of $\mathtt{CT}$-λ computing the iteration of maps on N. We define a normal term $\mathtt{It} : (N \to N), N, N \to N$ such that $\mathtt{It}(f, n, a) = f^n(a)$ for all $n \in N$. We have to to solve the equations $\mathtt{It}(f, a, 0) = a$ and $\mathtt{It}(f, a, \mathtt{S}(t)) = f(\mathtt{It}(f, a, t))$. We solve them with $\mathtt{It} = \lambda f, a, x.i[a, f, x]$ with $i[a, f, x] = (\mathtt{cond}x.(a, f(i[a, f, x])))(x)$.

The term is well-typed and regular by definition. We check the global trace condition. We mark the last unnamed argument N of $\mathtt{It}$. The mark moves to x in $i[a, f, x]$. Through a η-$\mathtt{rule}$-rule the mark moves to the unique unnamed argument N of $\mathtt{cond}x.(a, f(i[a, f, x]))$. The mark either moves to the name x in the context of a and there it stops, or it progresses and moves to x in the context of $f(i[f, a, x])$, then in the context of $i[f, a, x] : N$. Now we loop: if the path continues forever, then after infinitely many step the mark from which we started progresses infinitely many times, each time it crosses a $\mathtt{cond}$-rule.

5.3 The Interval Map

A third example. We simulate lists with two variables $\mathtt{nil} : \alpha$ and $\mathtt{cons} : N, \alpha \to \alpha$. We recursively define a notation for lists by $[] = \mathtt{nil}$, $a@l = \mathtt{cons}(a, l)$ and $[a, \vec{a}] = a@[\vec{a}]$. We add no elimination rules for lists, though, only the variables $\mathtt{nil}$ and $\mathtt{cons}$.

We will define a term $\mathtt{I}$ with one argument $f : N \to N)$ and three argument $a, x, y : N$, such that

$$\mathtt{I}(f, a, n, m) = [f^n(a), f^{n+1}(a), \ldots, f^{n+m}(a)]$$

for all $n, m \in N$. We have to to solve the recursive equations

$$\mathtt{I}(f, a, x, 0) = [\mathtt{It}(a, f, x)] \qquad \mathtt{I}(f, a, x, \mathtt{S}(t)) = \mathtt{It}(a, f, x)@\mathtt{I}(f, a, \mathtt{S}(x), t)$$

We solve them with $\mathtt{I} = \lambda f, a.v$, where $v : N, N \to N$, $v = \lambda x.\mathtt{cond}y.(\ [w],\ w@v(\mathtt{S}(x), y)\ )$ and $w = \mathtt{It}(f, a, x)$.

The term is well-typed and regular by definition. We check the global trace condition. We mark the last unnamed argument N of $\mathtt{I}$. The mark moves to the last unnamed argument N of $v : N, N \to N$. We unfold v to $\lambda x.\mathtt{cond}y.(\ \mathtt{cons}(w, \mathtt{nil}),\ \mathtt{cons}(w, v(\mathtt{S}(x), y))\ )$ The mark progresses and moves to the name y in the context of the body of the λ-abstraction, then to y in the context of $\mathtt{cons}(w, \mathtt{nil})$ or of $\mathtt{cons}(w, v(\mathtt{S}(x), y))$, then in the context of $\mathtt{nil}$ and stops, or in the context of $w = \mathtt{It}(f, a, x) : N$, for which we already checked the global trace condition, or in the context of $v(\mathtt{S}(x), y)) : \alpha$.

Then the mark moves from the named argument y of $v(\mathtt{S}(x)) : N \to N$ to the unique unnamed argument N of $v(\mathtt{S}(x))$, and eventually to the last argument of $v : N, N \to N$. From v we loop: after infinitely many step either we reached some w and we find some infinite progressing trace inside it, or the mark from which we started progresses infinitely many times through v. In both cases we have the global trace condition.

We have infinitely many nested β-reduction $(\lambda x. \ldots)(\mathtt{S}(x))$. We can remove all of them in a single step. Inside the β-redex number n we obtain a sub-term $w[\mathtt{S}(x)/x] \ldots [\mathtt{S}(x)/x]$ (substitution repeated n times). The result is $w[\mathtt{S}^n(x)] = \mathtt{It}(f, a, \mathtt{S}^n(x))$. The nested substitution produce new β-reductions $\mathtt{It}(f, a, \mathtt{S}^n(x)) = (\lambda x.i[f, a, x])(\mathtt{S}^n(x))$ for all $n \in N$. This is a non-regular term: we have infinitely many pairwise different sub-terms $x, \mathtt{S}(x), \mathtt{S}(\mathtt{S}(x)), \mathtt{S}(\mathtt{S}(\mathtt{S}(x))), \ldots$. We need infinitely many steps to normalize all $\mathtt{It}(f, a, \mathtt{S}^n(x))$ to $f^n(i[f, a, x])$, even if we allow to reduce all β, $\mathtt{cond}$-redexes at the same time. Also the normal form is not regular: it contains all terms $f^n(i[f, a, x])$ for $n \in N$, hence infinitely many pairwise different terms. These infinite sub-terms are of a particulary simple form, though. They are obtained by the repeating n times the assignment $z := f(z)$, then applying $z := i[f, a, x]$ once to the result.

Apparently, $\mathtt{I}$ is some term of $\mathtt{CT}$-λ which cannot be normalized in finite time, not even if we allow infinite parallel reductions without any "safety" restriction. The normal form is produced *only in the limit* and it is *not regular*. If we allow to reduce infinitely many nested β-redexes in one step, also the intermediate steps of the infinite reduction of $\mathtt{I}$ are not regular.

6 Weak normalization for closed terms of$\mathtt{CT}$-λ of type N

In this section we prove that every closed term of$\mathtt{CT}$-λ (that is, well-typed, regular and with the global trace condition) normalizes with finitely many "safe" steps to some numeral $n \in N$.

....
(Here we should insert our notes for the proof of weak normalization) ....

7 Uniqueness of normal form for closed terms of $\mathtt{CT}$-λ of type N

In this section we prove a weak form of confluence: normal form for all closed terms of $\mathtt{CT}$-λ of type N is unique.

By the weak normalization result proved in **S**6, we will deduce: for all for all closed terms t of $\mathtt{CT}$-λ of type N, there is some $n \in N$ such that there is a safe reduction $t \mapsto n$, and all normal form of t are equal to n.

We define a notion $\sim$ of equivalence for well-typed terms with the same type. Assume $t, u : A$ are *closed* terms of $\mathtt{CT}$-λ. (that is, well-typed, regular and with the global trace condition).

We define an equivalence $\sim_0$ for closed terms, by induction on the type A of t, u.

1. Assume $A = N$. Then $t \sim_0 u$ if and only if for all normal $t', u' : N$, if $t \mapsto t'$ and $u \mapsto u'$ then $t' = u'$.
2. Assume $A = B \to C$. Then $t \sim_0 u$ if and only if for all closed $b, c : B$ if $b \sim_0 c$ then $t(b) \sim_0 u(c)$.

We will prove that for all closed terms t of $\mathtt{CT}$-λ we have $t \sim_0 t$. If $A = N$ this means that the normal form of all closed terms of $\mathtt{CT}$-λ of type N is unique, which is our goal.

We first define $t \sim u$ for *any* terms of $\mathtt{CT}$-λ with the same type. $t \sim u$ if and only if for any two substitution $\sigma = [\vec{t}/\vec{x}]$ and $\tau = [\vec{u}/\vec{x}]$, if $\vec{t} \sim_0 \vec{u}$ and $\sigma(t)$, $\sigma(u)$ are closed then $\sigma(t) \sim_0 \sigma(u)$.

We will prove that for all terms t of $\mathtt{CT}$-λ we have $t \sim t$. In the case of closed terms t we can choose $\sigma = \tau =$ the empty sbustitution and we obtain $t \sim_0 t$, which is our goal.

Assume that $t : N$ is a closed term t of $\mathtt{CT}$-λ and $n \in N$.

1. We say that n is a value of t if $t \mapsto n$ (if t safely reduces to n).
2. We say that t is confluent if $t \sim_0 t$.

Assume $\vec{x} : \vec{A}$ and $t[\vec{x}] : \vec{D} \to N$. Assume $\vec{a} : \vec{A}, \vec{d} : \vec{D}$ and $\vec{b} : \vec{A}, \vec{e} : \vec{D}$ are vectors of closed terms of $\mathtt{CT}$-λ. We say that these four vectors are a counterexample to t if $\vec{a}, \vec{d} \sim_0 \vec{b}, \vec{e}$ and $\neg(t[\vec{a}](\vec{d}) \sim_0 t[\vec{b}](\vec{e}))$.

We have $t \sim t$ if and only if there is no counter-example for t.

By the weak normalization result proved in
Succ 6, all closed term t of $\mathtt{CT}$-λ of type N have a value. If t is confluent, then the value is unique. For confluent closed term t of $\mathtt{CT}$-λ of type N, "safe" reduction preserves the value. Indeed, if $t \mapsto u$ then u is a closed term of $\mathtt{CT}$-λ of type N, therefore u has a value m, which is also a value of t. By confluence of t we conclude that $n = m$.

If t is as above, with value n, and $t \mapsto \mathtt{S}(u)$, then $n > 0$, u is a confluent closed term t of $\mathtt{CT}$-λ of type N, and the value of u is $n - 1$. Indeed,if $u \mapsto u'$ and u' is normal, then $t \mapsto \mathtt{S}(u')$, therefore $\mathtt{S}(u') = n$ by confluence of t, and we conclude that $n > 0$ and $u' = n - 1$. Thus, the normal form of u is unique.

We will prove the following.

Theorem 7.1 *Assume t is any well-typed term such that $\neg(t \sim t)$. Then there is some infinite path $\pi = \{t_i | i \in N\}$ of t with some infinite sequence $\sigma = \{\sigma_i | i \in N\}$, with each σ_i counter-example for t_i, and with the value of σ compatible with the trace conditions of π.*

As a corollary from the theorem we will conclude: if t satisfies the global trace condition, then there is no such σ, therefore $t \sim t$, as we wished to show.

We prove that for any term $(t \sim t)$ with counter-example $\vec{a}, \vec{d} \sim_0 \vec{b}, \vec{e}$ and $\neg(t[\vec{a}](\vec{d}) \sim_0 t[\vec{b}](\vec{e}))$ we can find some immediate subterm t' and with a counter-example $\vec{a'}, \vec{d'} \sim_0 \vec{b'}, \vec{e'}$ and $\neg(t'[\vec{a'}](\vec{d'}) \sim_0 t'[\vec{b'}](\vec{e'}))$, *compatible with trace condition.*

8 Infinite reductions

For infinite terms we can define a notion of reduction contracting infinitely many redexes at the same time. In this section investigate this notion, we show that for recursive terms it is a recursive operation, and that the result is well-defined for the set `GlTrCon` of terms with the global trace condition. We also show that `GlTrCon` is closed under infinite reductions.

Assume $t \in \mathbb{A}$ and X is a set of redexes in t. We define an operation $\mapsto_X$ on each node t' of t, and its result $u \in \mathbb{A}$ by induction on the distance between t' and t.

If $t \not\in X$ and $t = 0, \mathtt{S}(u), x, \lambda x : Tu, f(u), \mathtt{cond}(f, g)$, we apply $\mapsto_X$ on all immediate subterms t' of t. If $t = (\lambda x : T.b)(u), \mathtt{cond}x.(f, g)(0), \mathtt{cond}x.(f, g)(\mathtt{S}(u))$, we contract t, respectively, to $b[u/x], f, g[u/x]$, then we apply $\mapsto_X$ to the result.

The operation $\mapsto_X$ loops if in some path of t we always find redexes in X. This is the case of $t = (\lambda x : T.t)(x)$ and of $u = \mathtt{cond}x.(0, u)(1)$ and of $X =$ all redexes of t, u respectively. In both terms, the unique infinite path of the term is made of redexes of X, and $\mapsto_X$ never provides a single symbol of the output.

If we want to consider infinite redexes in $\mathbb{A}$ we should allow undefined sub-terms inside a term. This is not the case for the set `GlTrCon` of terms with the global trace condition: we can show that `GlTrCon` is closed under $\mapsto_X$.

We first check that $\mapsto_X$ applied to `GlTrCon` produces no undefined sub-term.

Lemma 8.1 (No infinite rexed path) *If $t \in$ `GlTrCon` (if t has the global trace condition) then $\mapsto_X$ applied to t produces no undefined sub-term.*

Proof We have to prove that there is no path in t made only of redexes. Suppose a path in t contains an a redex $r = (\lambda x : T.b)(u)$ or $r = (\mathtt{cond}x.(f, g))(t)$, with $t = 0$ or $t = \mathtt{S}(u)$. We first check that no trace τ starting from r can progress in the next sub-term of the path. Indeed, τ can only continue through the left-hand-side of r, and if τ progresses, then the left-hand-side must be $\mathtt{cond}x.(f, g)$ and τ should move to the variable x bound by $\mathtt{cond}x.(f, g)$. The only way to obtain this is is that $t = x$ and r is obtained by an η-`rule`, and in this case r is no redex.

This implies that in a path π only made of redexes any trace τ cannot progress after its first point, and therefore it cannot infinitely progress and the term has no global trace condition.

We prove that `GlTrCon` is closed under $\mapsto_X$.

Lemma 8.2 *If $t \in$ `GlTrCon` and $t \mapsto_X u$ then $u \in\in$ `GlTrCon`*

Proof We prove that for any infinite path π in u there is some infinite path ρ in t such that for all traces τ of ρ there is some trace σ of π which is obtained by removing some points of τ which are either the first point or are not progress points. It will follow that if we have an infinitely progressing trace τ in ρ, then we have some infinitely progressing trace σ in π.

We follow π and we apply $\mapsto_X$. We find finitely many redexes which we reduce to some $v[t_n/x_n] \ldots [t_1/x_1]$ with v not redex. A single step in ρ through v corresponds to $n + 1$ steps in π, in such a way that a trace in π is restricted to a trace in ρ. The restriction removes no progresses point but at most the first.

If the path ρ continues in this way forever then we are done.

The other possibility is: π could move to some t_i which was an x_i in t. In this case we restart the path in t through t_i, while ρ continues in $t_i[t_{i-1}/x_{i-1}] \ldots [t_1/x_1]$. After finitely many steps ρ can continue through t_j for some $1 \leq j < i$, which was an x_j in t. This process can continue at most n times. Eventually ρ continues forever inside t_k for some $1 \leq k \leq n$. In this case there is a 1 to 1 correspondence between the trace in ρ and in t_k and the traces in π and in t_k.

9 Normalization and Fairness

. . . .
(Here we should insert our notes for the proof of strong normalization for safe reductions)
. . . .

There are terms requiring infinite reductions to normalize, but there are reductions normalizing in the limit. We conjecture that we can characterize reductions normalizing in the limit using the notion of "fair" reductions.

Definition 9.1 1. *We say that a node is in the k,* `cond`*-level if it has less than k nodes* `cond` *in its branch.*

2. *We say that an infinite reduction sequence is is fair for nodes of k,* `cond`*-level in the following case: for all $i \in N$, all $k > 0$, there is some $j \geq i$ such that at step j either all nodes in the k,* `cond`*-level are normal, or one of them is reduced at step j.*

Conjecture 1. For all $k \in N$, an infinite reduction sequence σ eventually stop reducing nodes in the k, `cond`-level.

Conjecture 1 should be proved by adapting the proof of strong normalization for safe reductions to a termination result on the first k-levels for all reductions, and assuming the same result for $k - 1$.

Conjecture 2. An infinite reduction sequence σ is normalizing in the limit if and only if for all $k \in N$, $k > 0$, σ is fair for the task of reducing nodes in the k, `cond`-level.

Conjecture 2 should follow in few steps from conjecture Conjecture 1.

10 Equivalence between cyclic and non-cyclic system T

In this section we prove that the set of total functionals on N definable in $\mathbb{T}$ and $\texttt{CT-}\lambda$ are the same.

The inclusion from $\mathbb{T}$ and $\texttt{CT-}\lambda$ is easy to prove. For any type T we can define a term $\texttt{Rec} : T, (N, T \to T) \to T$ such that $\texttt{Rec}(a, f, 0) = a$ and $\texttt{Rec}(a, fn + 1) = f(n, \texttt{Rec}(a, f, n))$, for all numeral $n \in \mathbb{N}$. The definition is $\texttt{Rec} = \lambda a, f.\texttt{rec}.$ with $\texttt{rec} = condx.(a, f(x, \texttt{rec}(x))) : N \to T$.

The opposite inclusion, from $\texttt{CT-}\lambda$ to $\mathbb{T}$, it has been proved by proof-theory for the combinatorial version of circular $\mathbb{T}$. For $\texttt{CT-}\lambda$, we will define instead an algorithm taking an infinite term in $\texttt{CT-}\lambda$, described as a finite circular tree, and returning a term in $\mathbb{T}$. Our translation can expand more than exponentially the size of the finite circular tree.

Assume $t : T$ is a cyclic term, represented as a cyclic tree with buds $t_1, \ldots, t_n$. We translate it to a term of $\mathbb{T}$. This is a first draft about how to do it.

1. We first move all buds to the same type and context, by adding dummy variables and dummy arguments. This operation preserves regularity and global trace condition. Now $t_1, \ldots, t_n$ all have context Γ and type A.
2. We merge all buds into the same term, defined by some u such that $u(i) = t_i$, for $i = 1, \ldots, n$, and $u(i) =$ some dummy term of type A otherwise. We replace each t_i with $u(i)$, for $i = 1, \ldots, n$. This operation preserves regularity and global trace condition. Now we have n buds, all are the same u with context Γ and type $N \to A$. Each bud b defines a partial bijection between the occurrence of N in its context and type $\Gamma \vdash N \to A$, and the occurrences of N in the context and type $\Gamma \vdash N \to A$ of its companion. We extend this partial bijection to any total bijection τ, depending on the but b.
3. We close the partial bijections defined by each bud by composition. The number of partial bijections can grow in an exponential way.
4. Assume we have m occurrences of N inside the context and type $\Gamma \vdash N \to A$ of u. We fix a permutation $\sigma : \{1, \ldots, m\}$ and we label them by variables $x_1, \ldots, x_n$ of $\mathbb{T}$, with x_i label of the argument with type N and number i. We will define a translation $t^\sigma \in \mathbb{T}$ of $t \in \texttt{CT-}\lambda$.
5. All traces move from u to any of the occurrences of u inside u. Some traces of some N in $\Gamma \vdash N \to A$ disappear, some other are moved to some other N, in an injective way. Two traces never merge. We label each trace in the bud u with the name x_i of the corresponding trace, if any. All those corresponding to no trace are labeled at random using the remaining variable names.
   At least one trace progresses, otherwise by repeating infinitely many times this step we would get a path with no progressing trace. The same is true for any combination of one or more movements from u to u.
6. At least one trace x_i progresses and it is not erased by any other trace. Otherwise we could follow a path in which each progress is erased in some new step, and so there is no infinite progressing trace. We use this trace as the main variable x_i of the recursion. In all steps, either x_i is constant

or decreases, and in at least one case it decreases. In all cases in which x_i decrease we use primitive recursion on x_i in $\mathbb{T}$, as main variable. In all other case, x_i is not removed, therefore it stays the same. We isolate the main variable x_j of the recursion for these steps, it is progressing therefore $j \neq i$. We use primitive recursion on x_j: this is the second variable of primitive recursion. We continue in this way and we define a primitive recursion in $\mathbb{T}$, with pairwise distinct indexes $x_{i_1} = x_i$, $x_{i_2} = x_j, \dots, x_{i_k}$ for some $k \geq 1$. We extend $x_{i_1}, \dots, x_{i_k}$ to $x_{i_1}, \dots, x_{i_n}$ in a random way: we defined in this way a permutation σ on $\{1, \dots, m\}$ by $\sigma(j) = i_j$ for $j \in \{1, \dots, m\}$ We define in this way a closed primitive recursive term $\lambda\vec{x}.t^\sigma \in \mathbb{T}$. Each bud u defining a permutation τ is replaced by $\mathtt{exch}_\tau(f)$. The term $\mathtt{exch}_\tau \in \mathbb{T}$ applies the permutation τ to the arguments of f, and during the recursive call f is replaced by $\lambda\vec{x}.u^\sigma$.

We claim that the infinite term $t^\sigma \in \mathbb{T}$ is equivalent to the cyclic recursive term $t \in \mathtt{CT}\text{-}\lambda$ we started from.