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 and simple types, instead of interpreting λ-abstraction through combinators and inserting a circular syntax afterwards. We introduce our circular syntax as a fixed point operator defined by cases. Our motivation for using binders instead of combinators is that this way of introducing circular syntax could be more familiar for researchers working in the field of Type Theory.

We prove the expected results for this circular simply typed λ-calculus, which we call CT-λ: every closed term of type N reduces to some numeral; strong normalization for all reductions sequences outside all fixed points; strong normalization for "fair" reductions; Church-Rosser property; and the equivalence with Gödel system $\mathbb{T}$.

Besides, terms of CT-λ are much shorter than the equivalent terms written in $\mathbb{T}$.

1 Introduction

We will introduce $\mathbb{A}$, a set of infinite λ-terms whose types are: the atomic type N, possibly type variables $\alpha, \beta, \ldots$, and all types $A \rightarrow B$ for any types A, B. The terms of $\mathbb{A}$ are possibly infinite trees representing expressions defined with 0,S,ap (application), variables, λ (the binder for defining maps), and cond, a fixed point binder operator for the arithmetic conditional (test on zero). If we have no type variables, the the trees in $\mathbb{A}$ represent partial functionals on N, provided we add reduction rules.

In this paper will consider the subsets CT-$\lambda \subset$ GTC, Reg $\subset$ WTyped, with WTyped the set of well-typed terms of $\mathbb{A}$. GTC is the subset of well-typed circular λ-terms satisfying a condition called the global trace condition, and they denote total functionals. Reg is the subset of terms which are regular trees (having finitely many subtrees), and therefore are possibly infinite terms which are finitely presented. CT-λ is GTC $\cap$ Reg, that is, it consists of all well-typed circular λ-terms satisfying satisfying the global trace condition and which are regural trees. We will prove that CT-λ is a decidable subset of Reg. CT-λ is a new circular version of Gödel system $\mathbb{T}$. Differently from all previous circular version of $\mathbb{T}$, our system CT-λ uses binders instead of combinators. Our goal is providing a circular syntax more familiar to researchers working in the field of Type Theory.

We will prove the expected results for the circular syntax CT-λ: strong normalization for reductions not inside any fixed point and Church-Rosser. We will prove normalization in the limit if we use reductions which are "fair": they can reduction *inside* fixed points, but they never forget entirely to reduce *outside* all fixed points. Besides, we will prove that the closed terms CT-λ (those without free *term* variables) represent exactly the total computable functionals 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, the subset WTyped of well-typed terms and a reduction relation for them.

We write $\mathbb{N}$ for the set of natural numbers: all $n \in \mathbb{N}$ are represented in $\mathbb{A}$ by expressions we call numerals.

Definition 2.1 (Terms of $\mathbb{A}$)

1. *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). As usual, we abbreviate* $\mathtt{ap}(t,u)$ *with $t(u)$.*

2. *t is an immediate subterm of $\mathtt{S}(t)$, f, g are immediate subterm of $\mathtt{cond}x.(f,g)$ and t, u are immediate subterm of $t(u)$. The subterm relation is the reflexive and transitive closure of the immediate subterm relation.*
3. *We write $\mathtt{SubT}(t)$ for the set of subterms of t.*
4. *We write $\mathtt{Tree}(t)$ for the tree of all chains $(t_1, \ldots, t_n)$, with $t_1 = t$ and t_{i+1} immediate subterm of t_i for all $(i+1) \leq n$. subterm*
5. *When $t = \mathtt{S}^n(0)$ for some natural number $n \in \mathbb{N}$ we say that t is a numeral. We write $\mathtt{Num}$ for the set of numeral.*

We use two different names for the operation $\mathtt{ap}(t, u)$: we call it $\mathtt{ap}$ when u is not a variable and η-rule when u is a variable.

$\mathtt{Num}$ is the representation inside $\mathbb{A}$ of the set $\mathbb{N}$ of natural numbers. All numeral are finite terms of Λ. All finite well-typed 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 $\mathbb{A}$. The set $\mathtt{SubT}(t) = \{t, 0\}$ of subterms of t is finite, therefore t is a regular term. However, $\mathtt{Tree}(t)$ is an infinite tree (it includes itself as a subtree). The finite branches of $\mathtt{Tree}(t)$ are all $(t, t, t, \ldots, 0)$, the unique infinite branch of $\mathtt{Tree}(t)$ is $(t, t, t, \ldots)$.

In order to define the type of a an infinite term, we first define the context for any term of $\mathbb{A}$.

Definition 2.2 (Types and Contexts of $\mathbb{A}$)

1. *The types of $\mathbb{A}$ are all types we can define with $\mathtt{N}$, an infinite list $\alpha, \beta, \ldots$ of type variables and $\rightarrow$. We call them simple types, types for short.*
2. *$\mathtt{Type}$ is the set of simples types.*
3. *A context of $\mathbb{A}$ is any finite list $\Gamma = (x_1 : A_1, \ldots, x_n : A_n)$ of pairwise distinct variables, each assigned to some type $A_1, \ldots, A_n \in \mathtt{Type}$. We write $\mathrm{FV}(\Gamma) = \{x_1, \ldots, x_n\}$.*
4. *If $\Gamma = (x_1 : A_1, \ldots, x_n : A_n)$, $\Gamma' = (x'_1 : A'_1, \ldots, x'_n : A'_{n'})$ are context of $\mathbb{A}$, then we write $\Gamma \sqsubseteq \Gamma'$ if Γ is a sub-list of Γ', that is, if there is a (unique) increasing map $f : \{1, \ldots, n\} \rightarrow \{1, \ldots, n'\}$ such that $x_i = x'_{f(i)}$ and $A_i = A'_{f(i)}$ for all $i \in \{1, \ldots, n\}$.*
5. *If Γ is a context of $\mathbb{A}$, then $\Gamma \setminus \{x\}$ is the context obtained by removing $x_i : A_i$ from Γ if $x_i = x$. If $x \notin \mathrm{FV}(\Gamma)$ then $\Gamma \setminus \{x\} = \Gamma$.*

From the context for a term we can define a context for each subterm of the term.

Definition 2.3 (Inherited Contexts of $\mathbb{A}$) *Given any context Γ, any $t \in \Lambda$ and any subterm $u \in \mathtt{SubT}(t)$, we define a unique inherited context for u, of the form Γ', Δ for some $\Gamma' \sqsubseteq \Gamma$ and some Δ.*

1. *t has inherited context Γ.*
2. *Any binder on x subtracts the variable x from the context of its last argument: if $t = \lambda x : T.u, \mathtt{cond}x.(f, g)$ has inherited context Γ', Δ, then u and g have context $(\Gamma', \Delta) \setminus \{x\}, x : T$, while f has inherited context Γ', Δ.*
3. *In any other case the context of a term and of the immediate subterm are the same: ff $t = \mathtt{S}(u), f(a)$ have inherited context Γ', Δ, then u, f, a have inherited context Γ', Δ.*

We abbreviate " inherited context from Γ" with context.

We define typing rules for terms of $\mathbb{A}$ and the subset $\mathtt{WTyped}$ of well-typed terms. The typing rules are the usual ones but for the conditional binder $\mathtt{cond}$, and for a fresh rule η-rule for typing the application $t(x)$ of a term to a variable x of type $\mathtt{N}$. η-rule corresponds to an η-expansion and it introduces a global variable name $x : T$ for the first argument of t. As we said, η-rule is but a particular case of $\mathtt{ap}$ rule. We prefer to use a separate name for η-rule because of the special role this rule has in this paper.

We have a a single structural rule $\mathtt{weak}$ for extending a context Γ to a context $\Gamma \sqsupseteq \Gamma'$. Variable permutation and variable renaming are conditionally derivable for the typing rules, and therefore are not included as rules.

Definition 2.4 (Typing rules of $\mathbb{A}$) *Assume $\Gamma = x_1 : A_1, \ldots, x_n : A_n$ is a context.*

1. $\mathtt{weak}$-*rule (Weakening). If* $\Gamma, \Delta \vdash t : T$, $\Gamma \sqsubseteq \Gamma'$ *then* $\Gamma' \vdash t : T$
2. η-*rule. If* $x \notin \mathrm{FV}(\Gamma)$ *is a variable and* $\Gamma \vdash f : \mathtt{N} \to T$ *then* $\Gamma, x : \mathtt{N} \vdash f(x) : T$.
3. $\mathtt{var}$-*rule. If* $x : A \in \Gamma$ *then* $\Gamma \vdash x : A$.
4. $\mathtt{ap}$-*rule. If* $\Gamma \vdash f : A \to B$ *and* $\Gamma \vdash a : A$ *and* a *is not a variable then* $\Gamma \vdash f(a) : B$.
5. λ-*rule. If* $\Gamma, x : A \vdash b : B$ *then* $\Gamma \vdash \lambda x : A.b : A \to B$.
6. 0-*rule.* $\Gamma \vdash 0 : \mathtt{N}$
7. $\mathtt{S}$-*rule. If* $\Gamma \vdash t : \mathtt{N}$ *then* $\Gamma \vdash \mathtt{S}(t) : \mathtt{N}$.
8. $\mathtt{cond}$-*rule. If* $\Gamma \vdash f : T$ *and* $\Gamma, x : T \vdash g : T$ *then* $\Gamma \vdash \mathtt{cond}x.(f, g) : \mathtt{N} \to T$.

We abbreviate $\emptyset \vdash t : A$ *with* $\vdash t : A$.

From the typing rules we define the proofs that a term is well-typed.

Definition 2.5 (Well-typed term of $\mathbb{A}$) *We write* $\Pi : \Gamma \vdash t : A$, *and we say that* Π *is a proof of* $\Gamma \vdash t : A$ *if* $\Pi : \mathtt{SubT}(t) \to \mathtt{Type}$ *is an assignment of one type* $B = \Pi(\pi)$ *to each subterm chain* $\pi \in \mathtt{SubT}(t)$, *in such a way that:*

1. $\Pi(\mathtt{nil}) = A$
2. *Assume* π *is a subterm chain from* t *to* u, *if* $u_1, \ldots, u_n$ *are the immediate subterms of* u, $\Gamma_1, \ldots, \Gamma_n, \Gamma$ *are the inherited contexts, and* $B = \Pi(\pi)$, $B = \Pi(\pi \star u_1)$, $\ldots$, $B = \Pi(\pi \star u_n)$. *then for some* $\Gamma' \sqsubseteq \Gamma$ *we have*
$$\dfrac{\dfrac{\Gamma_1 \vdash u_1 : B_1 \ldots \Gamma_n \vdash u_1 : B_n}{\Gamma' \vdash u : B}\,(\mathtt{r})}{\Gamma \vdash u : B}\,(\mathtt{weak})$$
for some $\mathtt{r} \in \{0, \mathtt{S}, \mathtt{var}, \mathtt{ap}, \lambda, \mathtt{cond}\}$ *(for some non-weakening rule* $\mathtt{r}$*).*

We introduce the type judgment for a term.

1. $\Gamma \vdash t : A$ *is true if and only if* $\Pi : \Gamma \vdash t : A$ *for some* Π.
2. $t \in \mathbb{A}$ *is well-typed if* $\Gamma \vdash t : A$ *for some* Γ, A.
3. $\mathtt{WTyped}$ *is the set of well-typed* $t \in \mathbb{A}$.

1. Some term in $\mathbb{A}$ has no type, like $0(0)$. Some terms have more than one type. For instance if $t = \mathtt{cond}x.(t, t)(0)$ we can prove $x : \mathtt{N} \vdash t : A$ for all types $A \in \mathtt{Type}$. A term with two types has the leftmost branch infinite, as it is the case for t above.
2. We will consider terms, those satisfying the global trace condition, for which typing when it exists it is unique. We will prove that term with the global trace condition has the leftmost branch finite and therefore has a unique tye.

As we said, η-rule is a particular case of $\mathtt{ap}$. When we substitute $x : \mathtt{N}$ with some $\Gamma \vdash a : \mathtt{N}$ which is not a variable, then we replace η-rule with $\mathtt{ap}$-rule.

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

Our goal is to provide a set of well-formed term for $\mathbb{A}$ and interpret them as partial functionals. Some terms, those satisfying the global trace condition (to be introduced later) will be total functionals. Our first step is to provide reduction rules for $\mathbb{A}$.

Definition 2.6 (reduction rules for $\mathbb{A}$)

1. $\mapsto_\beta$: $(\lambda x : A.b)(a) \mapsto_\beta b[a/x]$
2. $\mapsto_{\mathtt{cond}}$: $\mathtt{cond}x.(f, g)(0) \mapsto_{\mathtt{cond}} f$ *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* $\mathtt{cond}x.(f, g)$.
5. *A term is safe-normal if all its redexes (if any) are inside some* $\mathtt{cond}x.(f, g)$.

As example. If $u = \texttt{cond}x.(0, (\lambda x.u)(z))$ is as above, then u is safe-normal, because all redexes in u are of the form $(\lambda x.u)(z)$ and inside a cond. However, the tree form of u has the following branch:

$$u, \quad (\lambda x.u)(z), \quad \lambda x.u, \quad u, \quad \ldots$$

This branch is cyclic, infinite, and it includes infinitely many β-redexes.

The reason for forbiding reductions inside $\texttt{cond}x.(f, g)$ 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. Therefore reductions for cond can can easily loop, and they are "unsafe". For this reason, we first consider the minimum possible of reductions $\texttt{cond}x.(f, g)(0) \mapsto_{\texttt{cond}} f$ and $\texttt{cond}x.(f, g)(\texttt{S}(t)) \mapsto_{\texttt{cond}} g[t/x]$: those on maximal cond-expression. We consider no reduction inside the arguments f, g of cond. In a second moment, by adding a restriction of *fairness* on reduction strategies, we will be able to recover strong normalization also for most "unsafe" reductions.

Some examples of reduction.

1. An example of reduction of a term $v(n)$ to a normal form. If n is any numeral and $v = \texttt{cond}x.(0, (\lambda x.v(x)))$, then $v(n) : \texttt{N}$ has infinitely many β-redexes inside cond, therefore infinitely many unsafe reductions are possible. There are only finitely many safe reduction from $v(n)$ instead: $v(n)$ cond-reduces to $(\lambda x.v(x))(n-1)$, this latter β-reduces to $v(n-1)$, then we loop: $v(n-1)$ reduces to $v(n-2)$ in one cond-step and one β-step and so forth. After n cond-reductions and n β-reductions we get $v(0)$. With one last cond-reduction we get 0 and we stop.
2. The term $v(n)$ above is also an example of a strongly normalizing term. There is the unique reduction sequence from $v(n)$, because we cannot cond-reduce $\lambda x.v(x)$ before assigning either 0 or $\texttt{S}(u)$ to x. Thus, all reductions sequences from $v(n)$ terminate in $2n+1$ steps to the normal form 0.

3 The trace of the λ-terms

Total λ-terms will be well-typed terms $\Gamma \vdash t : A$ satisfying a condition call *global trace condition* for the canonical proof $\Pi : \Gamma \vdash t : A$. In order to define the global trace condition, 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 argument types* and index of atomic types for a term.

First, an example. Assume $t : \{x_1 : A_1, x_2 : A_2\} \vdash B_3 \to \texttt{N}$. Then the list of argument types of t is A_1, A_2, B_3, A_1, A_2 are named arguments with names x_1, x_2, and B_3 is an unnamed argument. Remark that for an open term t we list as "argument types" also the types of the free variables. We motivate our terminology: in a sense, t is an abbreviation of the closed term $t' = \lambda x_1 : A_1, x_2 : A_2.t : (A_1, A_2, B_3 \to \texttt{N})$, and the argument types of t' are in fact A_1, A_2, B_3. The index of an atomic argument of t is any $j \in \{1, 2, 3\}$ such that $A_j = \texttt{N}$ or $B_j = \texttt{N}$ respectively.

Definition 3.1 (List of argument types of a term) *Assume that* $\vec{A} = A_1, \ldots, A_n$, $\vec{B} = B_{n+1}, \ldots, B_{n+m}$, $\Gamma = \{\vec{x} : \vec{A}\}$, *and* $t : \Gamma \vdash \vec{B} \to \texttt{N}$.

1. *The list of argument types of t is $\vec{C} = \vec{A}, \vec{B}$, the list of types of free variables and arguments of the type $\vec{B} \to \texttt{N}$ of t.*
2. *$A_1, \ldots, A_n$ are named arguments with names $x_1, \ldots, x_n$, and $B_{n+1}, \ldots, B_{n+m}$ are unnamed arguments.*
3. *An index of an atomic argument of t is any $j \in \{1, \ldots, n+m\}$ such that $C_j = \texttt{N}$.*

We now define the atom correspondence between arguments of type N of subterms of t in a proof $\Pi : t : \Gamma \vdash A$ of $\mathbb{A}$. The definition of atom correspondence for a syntax including binders is the main contribution of this paper.

Two arguments of type N are in correspondence if and only if they receive the same global input: local input are ignored. In many cases two corresponding argument types have the same index, but if we insert or remove free variables or arguments the index may change. Before providing the general definition, we discuss these special cases through examples.

We draw in the same color two arguments of type N which are in correspondence.

3.1 Some correspondence examples

Example 3.1 An example of atom correspondence for some instance of the weakening rule.

$$\frac{x_1 : \mathtt{N}, x_2 : \mathtt{N} \vdash t : \mathtt{N} \to \mathtt{N}}{x_1 : \mathtt{N}, x_2 : \mathtt{N}, x_3 : \mathtt{N} \vdash t : \mathtt{N} \to \mathtt{N}} \ (\mathtt{weak})$$

Remark that the type N of the variable x_3 (colored in **old gold**) , introduced by weakening, is in correspondence with no type in the rule weak.

Example 3.2 An example of atom correspondence for the rule ap. We assume that a is *not* a variable.

$$\frac{x_1 : \mathtt{N}, x_2 : \mathtt{N} \vdash f : \mathtt{N}, \mathtt{N} \to \mathtt{N} \qquad x_1 : \mathtt{N}, x_2 : \mathtt{N} \vdash a : \mathtt{N}}{x_1 : \mathtt{N}, x_2 : \mathtt{N} \vdash f(a) : \mathtt{N} \to \mathtt{N}} \ (\mathtt{ap})$$

Remark that the first argument of f (colored in **old gold**) is in correspondence with no argument in the rule ap. The reason is that in the term $f(a)$, the first argument of f receives a value from the value a which is local to the term $f(a)$. However, the first argument of f can be in correspondence with some argument higher in the proof.

Example 3.3 An example of atom correspondence for the rule cond.

$$\frac{x_1 : \mathtt{N}, x_2 : \mathtt{N} \vdash f : \mathtt{N} x_1 : \mathtt{N}, x_2 : \mathtt{N}, x : \mathtt{N} \vdash g : \mathtt{N}}{x_1 : \mathtt{N}, x_2 : \mathtt{N} \vdash \mathtt{cond} x.(f, g) : \mathtt{N} \to \mathtt{N}} \ (\mathtt{cond})$$

Remark that the first argument of f (colored in **old gold**) is in correspondence the type of the free variable x in the premise for g, but it is in correspondence with no argument in the in the premise for f.

3.2 A formal definition of atom correspondence

We define an injection

$$\mathtt{ins} : \mathtt{N}, \mathtt{N} \to \mathtt{N}$$

by $\mathtt{ins}(a, x) = x + 1$ if $x \geq a + 1$, and $\mathtt{ins}(a, x) = x$ if $x \leq a$. The role of ins is inserting one space for a new index of argument type.

Definition 3.2 (Atom correspondence in a proof of $\mathbb{A}$) *Assume $\vec{A} = A_1, \ldots, A_n$, $\vec{B} = B_1, \ldots, B_m$, $\Gamma = \vec{x} : \vec{A}$, and $\Gamma \vdash t : \vec{B} \to \mathtt{N}$.*

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* weak *from t' then $k = \mathtt{ins}(a, k)$.*
2. *if $t = f(a)$ and a is not a variable and $t' = f$ then $k' = \mathtt{ins}(n, k)$. If $t' = a$ then $k' = k$ and $k' \leq n$.*
3. *if $t = \mathtt{cond} x(f, g)$ and $t' = f$ then $k' = \mathtt{ins}(n, k)$. If $t' ga$ then $k' = k$.*

We require $k = k'$ in all other cases, which are: S*, λ, η-rule.*

If we move up in a η-rule the type of one free variable corresponds to the type of one argument. In a λ-rule it is the other way round. The index of two corresponding argument types is the same for both rules. Below we include some examples. We draw in the same color two arguments of type N which are in correspondence.

Example 3.4 Atom correspondence for η-rule. We assume that x is a variable.

$$\frac{x_1 : \mathtt{N}, x_2 : \mathtt{N} \vdash f : \mathtt{N}, \mathtt{N} \to \mathtt{N}}{x_1 : \mathtt{N}, x_2 : \mathtt{N}, x : \mathtt{N} \vdash f(x) : \mathtt{N} \to \mathtt{N}} \ (\mathtt{ap})$$

Remark that the first argument of f (colored in **old gold**) in the premise is in correspondence with the last variable type in the conclusion of the rule.

Example 3.5 Atom correspondence for λ-rule. We assume that x is a variable.

$$\dfrac{x_1 : \mathtt{N}, x_2 : \mathtt{N}, x : \mathtt{N} \vdash b : \mathtt{N}}{x_1 : \mathtt{N}, x_2 : \mathtt{N} \vdash \lambda x : \mathtt{N}.b : \mathtt{N} \to \mathtt{N}}\ (\mathtt{ap})$$

Remark that the first argument of $\lambda x : \mathtt{N}.b$ (colored in **old gold**) in the conclusion is in correspondence with the last variable type in the premise of the rule.

The correspondence on atoms in a proof $\Pi : \Gamma \vdash$ defines a graph $\mathtt{Graph}(\Pi)$ whose nodes are all pairs (k, u), with u subterm of t and k index of some atomic argument of u. We define a trace as a (finite or infinite) path in the graph $\mathtt{Graph}(\Pi)$.

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 $\pi = t_0, \ldots, t_n, \ldots$ is a branch of t, finite or infinite. A finite or infinite trace τ of π is a list $\tau = (k_m, t_m), \ldots, (k_n, t_n), \ldots$ such that for each $i = m, \ldots, n, \ldots$:

(1) k_i is the index of an atomic type of t_i

(2) if $i + 1$ is an index of τ then k_{i+1}, t_{i+1} is in correspondence with k_i, t_i in Π.

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* unnamed *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`, which is an infinite term defined by $\mathtt{Sum} = \lambda x : \mathtt{N}.\mathtt{cond}z.(x, \mathtt{S}(\mathtt{Sum}(x)(z)))$. This term is regular with subterms

$$\mathtt{Sum}, \quad \mathtt{cond}z.(x, \mathtt{S}(\mathtt{Sum}(x)(z))), \quad x, \quad \mathtt{S}(\mathtt{Sum}(x)(z)), \quad \mathtt{Sum}(x)(z), \quad \mathtt{Sum}(x)$$

This term is well-typed by the following circular derivation with a back edge from the ($\dagger$) above to the ($\dagger$) below. The only infinite path contains a progressing trace, a sequence of `N`'s colored in **old gold**.

$$\dfrac{\dfrac{\dfrac{}{x : \mathtt{N} \vdash x : \mathtt{N}}\ \mathtt{var} \qquad \dfrac{\dfrac{\dfrac{\vdash \mathtt{Sum} : \mathtt{N} \to \mathtt{N} \to \mathtt{N}\ (\dagger)}{x : \mathtt{N} \vdash \mathtt{Sum}(x) : \mathtt{N} \to \mathtt{N}}\ \eta}{x : \mathtt{N}, z : \mathtt{N} \vdash \mathtt{Sum}(x)(z) : \mathtt{N}}\ \eta}{x : \mathtt{N}, z : \mathtt{N} \vdash \mathtt{S}(\mathtt{Sum}(x)(z)) : \mathtt{N}}\ \mathtt{S}}{x : \mathtt{N} \vdash \mathtt{cond}z.(x, \mathtt{S}(\mathtt{Sum}(x)(z))) : \mathtt{N} \to \mathtt{N}}\ \mathtt{cond}}{\vdash \mathtt{Sum} : \mathtt{N} \to \mathtt{N} \to \mathtt{N}\ (\dagger)}\ \lambda$$

5.2 The Iterator

A second example. We define a term $\mathtt{It}$ of $\mathtt{CT}\text{-}\lambda$ computing the iteration of maps on $\mathtt{N}$. We define a normal term $\mathtt{It} : (\mathtt{N} \to \mathtt{N}), \mathtt{N}, \mathtt{N} \to \mathtt{N}$ such that $\mathtt{It}(f, a, n) = f^n(a)$ for all $n \in \mathtt{N}$. 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 $\mathtt{It} = \lambda f, a, x.I$ with $I = (\mathtt{cond}x.(a, f(I)))(x)$.

The term is well-typed and regular by definition. We check the global trace condition. We mark the last unnamed argument $\mathtt{N}$ of $\mathtt{It}$. The mark moves to x in I. Through a η-rule-rule the mark moves to the unique unnamed argument $\mathtt{N}$ of $\mathtt{cond}x.(a, f(I))$. 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 $f(I)$, then in the context of $I : \mathtt{N}$. 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 $\mathtt{cons} : \mathtt{N}, \alpha \to \alpha$. 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 $f : \mathtt{N} \to \mathtt{N})$ and three argument $a, x, y : \mathtt{N}$, such that

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

for all $n, m \in \mathtt{N}$. 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 $v : \mathtt{N}, \mathtt{N} \to \mathtt{N}$, $v = \lambda x.\mathtt{cond}y.(\ [w],\ w@v(\mathtt{S}(x), y)\ )$ 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 $\mathtt{N}$ of $\mathtt{I}$. The mark moves to the last unnamed argument $\mathtt{N}$ of $v : \mathtt{N}, \mathtt{N} \to \mathtt{N}$. We unfold v to $\lambda x.\mathtt{cond}y.(\ [w],\ [w]@v(\mathtt{S}(x), y))\ )$ 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 $[w]$ or of $[w]@v(\mathtt{S}(x), y))$, then in the context of $\mathtt{nil}$ and stops, or in the context of $w = \mathtt{It}(f, a, x) : \mathtt{N}$, for which we already checked the global trace condition, or in the context of $v(\mathtt{S}(x), y)) : \mathtt{N}$.

Then the mark moves from the named argument y of $v(\mathtt{S}(x), y) : \mathtt{N}$ to the unique unnamed argument $\mathtt{N}$ of $v(\mathtt{S}(x))$, and eventually to the second argument of $v : \mathtt{N}, \mathtt{N} \to \mathtt{N}$. From v we loop: after infinitely many step either we reached some $w = \mathtt{It}(f, a, x)$ and we find some infinite progressing trace inside it, or the mark from which we started progresses infinitely many times through the $\mathtt{cond}$-rule inside 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 k we obtain a sub-term $w[\mathtt{S}(x)/x] \ldots [\mathtt{S}(x)/x]$ (substitution repeated k times). The result is $w[\mathtt{S}^k(x)] = \mathtt{It}(f, a, \mathtt{S}^k(x))$. The nested substitution produce new β-reductions $\mathtt{It}(f, a, \mathtt{S}^k(x)) = (\lambda x.I)(\mathtt{S}^k(x))$ for all $k \in \mathtt{N}$. This is a non-regular term: we have infinitely many pairwise different sub-terms $x, \mathtt{S}(x), \mathtt{S}^2(x), \mathtt{S}^3(x), \ldots$. We need infinitely many steps to normalize all $\mathtt{It}(f, a, \mathtt{S}^k(x))$ to $f^k(I)$, even if we allow to reduce all $\beta, \mathtt{cond}$-redexes at the same time. Also the normal form is not regular: it contains all terms $f^k(I)$ for $k \in \mathtt{N}$, hence infinitely many pairwise different terms. These infinite sub-terms are of a particulary simple form, though. They are obtained by the repeating k times the assignment $z := f(z)$, then applying $z := I$ once to the result.

Apparently, $\mathtt{I}$ is some term of $\mathtt{CT}\text{-}\lambda$ 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 Subject reduction for terms of $\mathtt{CT}\text{-}\lambda$ of type $\mathtt{N}$

....
(Here we should fill this part) ....

7 Subject reduction for terms of $\mathtt{CT}$-λ of type $\mathtt{N}$

. . . .
(Here we should fill this part) . . . .

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

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 $n \in \mathtt{N}$. In the following, we explicitly write $t[x_1, \ldots, x_n]$, when each free variable in t is some x_i, and, under this notation, also write $t[a_1, \ldots, a_n]$ instead of $t[a_1/x_1, \ldots, a_n/x_n]$. We recall that we denote with $\mathbb{N}$ the set of all numerals, namely all terms of the form $\mathtt{S}^n(0)$ for some $n \in \mathtt{N}$.

We define total terms t of type $\mathtt{N}$, the closed terms of type $\mathtt{N}$ evaluating in at least one reduction path to a numeral, and we define values, which are numerals (hence closed term). If $t : A \to B$ is a closed term, then total terms and values coincide, both are the terms mapping values to total terms. Values are always closed terms. An open term is total if all substitutions of free variables with values produce a closed total term. This is the formal definition:

Definition 8.1 (Values and total) *We define values and closed total terms on values by induction on types.*

1. *A closed term $t : \mathtt{N}$ is closed total if and only if $t \mapsto_{\mathtt{safe}}{}^* u$ for some $u \in \mathbb{N}$.*
2. *A closed term $t : \mathtt{N}$ is a value if and only if $t \in \mathbb{N}$.*
3. *A closed term $t : A \to B$ is total if and only if $t(a)$ is total for any value $a : A$.*
4. *Closed terms and values of type $t : A \to B$ coincide.*
5. *A term $t[\vec{x}] : C$, whose all free variables are $\vec{x} : \vec{A}$, is total if and only if $t[\vec{a}]$ is total for any values $\vec{a}$ of type $\vec{A}$.*

Closed total terms are closed by safe reductions.

Lemma 8.2 *1. Let $t : A$ be a closed term and $t \mapsto_{\mathtt{safe}} u$. If u is total, then so is t.*

2. Let $t[\vec{x}] : \vec{A} \to \mathtt{N}$ be a term, whose all free variables are $\vec{x} : \vec{B}$, and $\vec{u} : \vec{B}$ and $\vec{a} : \vec{A}$ be closed values. If $t[\vec{u}]\vec{a} : \mathtt{N}$ is total, then $t[\vec{x}]$ is total.

Proof 1. *The claim 1.* is shown by induction on A.

(1) We first show the *base case*, namely when A is $\mathtt{N}$. By the assumption and the subject reduction property, u has type A and is closed total. Then we have $t \mapsto_{\mathtt{safe}} u \mapsto_{\mathtt{safe}}{}^* n$ for some $n \in \mathtt{N}$. Hence $t : \mathtt{N}$ is total.

(2) We show the *induction case*, namely when A is $A_1 \to A_2$. Take arbitrary closed value $a : A_1$. Then we have $t(a) \mapsto_{\mathtt{safe}} u(a)$ and $u(a) : A_2$ is total by the assumption and the subject reduction property. Hence by the induction hypothesis $t(a)$ is total. We obtain that $t : A_1 \to A_2$ is total.

2. The claim 2. is shown by induction on $|\vec{A}|$.

(1) The *base case* $|\vec{A}| = 0$ is immediately shown by the definition.

(2) We show the *induction case*. Let $\vec{A} = A_0\vec{A'}$. Take arbitrary values $\vec{u} : \vec{B}$, $\vec{a'} : \vec{A'}$, and $a_0 : A_0$. By the assumption, we have $t[\vec{u}]a_0\vec{a'} : \mathtt{N}$ is closed total for all values $\vec{a'}$. Then $t[\vec{u}]a_0 : \vec{A'} \to \mathtt{N}$ is total by the induction hypothesis on $\vec{A'} \to \mathtt{N}$. By definition $t[\vec{u}] : \vec{A} \to \mathtt{N}$ is closed total, and so by definition $t[\vec{x}]$ is total, as we expected.

Let Π be a proof and e be a node of Π. We write $\Pi(e)$ for $t : \Gamma \vdash A$ at the node e in Π.

Theorem 8.3 *Assume $\Pi : \Gamma \vdash t : A$. If t is not total, then there is some infinite path in Π with no infinite progressing trace.*

Proof Assume that t is not total and $t : \vec{x} : \vec{D} \vdash \vec{A} \to \mathtt{N}$ has a proof Π (we show a contradiction). By 2. of Lemma 8.2, there exist closed values $\vec{a} : \vec{A}$ and $\vec{d} : \vec{D}$ such that $t[\vec{d}]\vec{a}$ is not total. We inductively construct $(e_i, \vec{d_i}, \vec{a_i})$ for each natural number i, such that

(i) e_i is a node of Π, where $t_i : \vec{x_i} : \vec{D_i} \vdash \vec{A_i} \to \mathtt{N}$ is at the node e_i in Π, and e_{i+1} is a child node of e_i in Π;

(ii) t_i is not total;

(iii) $\vec{d_i} : \vec{D_i}$ and $\vec{a_i} : \vec{A_i}$ are closed values such that $t_i[\vec{d_i}]\vec{a_i}$ is not total;

(iv) there is a trace compatible connection from $(\vec{d_i}; \vec{a_i})$ to $(\vec{d_{i+1}}; \vec{a_{i+1}})$, for any $i \geq 0$, in Π. Moreover, if i is a progress point, namely t_i is of the form $\mathtt{cond}x.(f, g)$ and t_{i+1} is g, then $\vec{a_i} = \mathtt{S}(m')\vec{a'}$, $\vec{d_{i+1}} = \vec{d_i}m'$, and a trace passes from $\mathtt{S}(m')$ to m'.

We first define $(e_0, \vec{d_0}, \vec{a_0})$ by $(r, \vec{d}, \vec{a})$, where r is the root node of Π. Then we have $t_0 = t$, (ii) and (iii). Next, assume that $(e_i, \vec{d_i}, \vec{a_i})$ is already constructed. Then we define $(e_{i+1}, \vec{d_{i+1}}, \vec{a_{i+1}})$ by the case analysis of the last rule for the node e_{i+1} in Π.

The case of $\mathtt{struct}(f)$, namely $\Pi(e_i) = t[y_{f(1)}/x_1, \ldots, y_{f(n)}/x_n] : \Delta \vdash \vec{A_i} \to \mathtt{N}$ is obtained from $t : \Gamma \vdash A$, where $\Gamma = x_1 : C_1, \ldots, x_n : C_n$, $\Delta = y_1 : B_1, \ldots, y_m : B_m$, $f : \{1, \ldots, n\} \to \{1, \ldots, m\}$ is an injection, and $C_i = B_{f(i)}$ for all $1 \leq i \leq n$. By the induction hypothesis and (b), $t[y_{f(1)}/x_1, \ldots, y_{f(n)}/x_n][d_{i,f(1)}/y_{f(1)}, \ldots, d_{i,f(n)}/y_{f(n)}]\vec{a_i} : \mathtt{N}$ is not total, where $\vec{d_i} = d_{i,1} \ldots d_{i,m}$. Then define e_{i+1} as the unique parent node of e_i, and also define $\vec{d_{i+1}}$ and $\vec{a_{i+1}}$ by $d_{i,f(1)} \ldots d_{i,f(n)}$ and $\vec{a_i}$, respectively. We obtain (i), (ii), and (iii) for $i+1$, as expected. We also have (iv) since the connection, determined by f, from $(d_{i,1} \ldots d_{i,m}; \vec{a_i})$ to $(\vec{d_{i+1}}; \vec{a_{i+1}}) = (d_{i,f(1)}, \ldots d_{i,f(n)}; \vec{a_i})$ is trace compatible. The case of η-rule, namely $\Pi(e_i) = f[\vec{x}](x) : \Gamma, x : N \vdash \vec{A} \to \mathtt{N}$ is obtained from $f[\vec{x}] : \Gamma \vdash N \to \vec{A} \to \mathtt{N}$, where $x \notin \Gamma$. By the induction hypothesis, $f[\vec{d'}]d\vec{a_i} : \mathtt{N}$ is not total, where $\vec{d_i} = \vec{d'}d$. Define e_{i+1} as the unique parent node of e_i, and also define $\vec{d_{i+1}}$ by $\vec{d'}$ and define $\vec{a_{i+1}}$ by $d\vec{a_i}$. We obtain (i), (ii), and (iii) for $i+1$, as expected. We also have (iv) since the connection from $(\vec{d_i}; \vec{a_i}) = (\vec{d'}d; \vec{a_i})$ to $(d_{i+1}; \vec{a_{i+1}}) = (\vec{d'}; d\vec{a_i})$ is trace compatible.

The case of 0-rule, namely $\Pi(e_i) = 0 : \Gamma \vdash \mathtt{N}$, is not the case since $t_i = 0$ is total.

The case of S-rule, namely $\Pi(e_i) = \mathtt{S}(t_{i+1}) : \Gamma \vdash \mathtt{N}$ is obtained from $t_{i+1} : \Gamma \vdash \mathtt{N}$. By the induction hypothesis, $\vec{a_i}$ is empty, and $\mathtt{S}(t_{i+1}[\vec{d_i}]) : \mathtt{N}$ is not total. Then, by the definition, $t_{i+1}[\vec{d_i}]$ is not total. Define e_{i+1} as the unique parent node of e_i, and also define $\vec{d_{i+1}}$ and $\vec{a_{i+1}}$ by d_i and empty, respectively. We obtain (i), (ii), (iii), and (iv) for $i+1$, as expected.

The case of ap-rule, namely $\Pi(e_i) = t[\vec{x}](u[\vec{x}]) : \Gamma \vdash \vec{A} \to \mathtt{N}$ is obtained from $t[\vec{x}] : \Gamma \vdash B \to \vec{A} \to \mathtt{N}$ and $u[\vec{x}] : \Gamma \vdash B$. By the induction hypothesis, $t[\vec{d_i}](u[\vec{d_i}])\vec{a_i} : \mathtt{N}$ is not total. We first consider the subcase that $u[\vec{d_i}] : B$ is total. We define a value $a' : B$ by $n \in \mathbb{N}$ such that $u[\vec{d_i}] \mapsto_{\mathtt{safe}}{}^* n$ if $B = \mathtt{N}$, by $u[\vec{d_i}]$ otherwise. Then define e_{i+1} by the parent node whose term is $t[\vec{x}]$, and define $\vec{d_{i+1}} = \vec{d_i}$ and $\vec{a_{i+1}} = a'\vec{a_i}$. Using Lemma 8.2, we obtain (i), (ii), (iii) for $i+1$, as expected. We also have (iv) since the connection from $(\vec{d_i}; \vec{a_i})$ to $(\vec{d_{i+1}}; \vec{a_{i+1}}) = (\vec{d_i}; a'\vec{a_i})$ is trace compatible. Next we consider the subcase that $u[\vec{d_i}] : B$ is not total. Then there is a sequence of values $\vec{a'}$ such that $u[\vec{d_i}]\vec{a'} : \mathtt{N}$ is not total. Define e_{i+1} by the parent node whose term is $u[\vec{x}]$, and define $\vec{d_{i+1}} = \vec{d_i}$ and $\vec{a_{i+1}} = \vec{a'}$. We obtain (i), (ii), (iii) for $i+1$, as expected. We also have (iv) since the connection from $(\vec{d_i}; \vec{a_i})$ to $(\vec{d_{i+1}}; \vec{a_{i+1}}) = (\vec{d_i}; \vec{a'})$ is trace compatible.

The case of λ-rule, namely $\Pi(e_i) = \lambda x.(t_{i+1}[\vec{x}, x]) : \Gamma \vdash A \to \vec{A} \to \mathtt{N}$ is obtained from $t_{i+1}[\vec{x}, x] : \Gamma, x : A \vdash \vec{A} \to \mathtt{N}$. By the induction hypothesis, $(\lambda x.(t_{i+1}[\vec{d_i}, x]))a\vec{a'} : \mathtt{N}$ is not total, where $\vec{a_i} = a\vec{a'}$. Then, by Lemma 8.2, $t_{i+1}[\vec{d_i}, a]\vec{a'}$ is not total. Define e_{i+1} as the unique parent node of e_i, and also define $\vec{d_{i+1}} = \vec{d_i}a$ and $\vec{a_{i+1}} = \vec{a'}$. We obtain (i), (ii), (iii) for $i+1$, as expected. We also have (iv) since the connection from $(\vec{d_i}; \vec{a_i}) = (\vec{d_i}; a\vec{a'})$ to $(\vec{d_{i+1}}; \vec{a_{i+1}}) = (\vec{d_i}a; \vec{a'})$ is trace compatible.

The case of cond-rule, namely $\Pi(e_i) = \mathtt{cond}x.(f[\vec{x}], g[\vec{x}, x]) : \Gamma \vdash \mathtt{N} \to \vec{A} \to \mathtt{N}$ is obtained from $f[\vec{x}] : \Gamma \vdash \vec{A} \to \mathtt{N}$ and $g[\vec{x}, x] : \Gamma, x : \mathtt{N} \vdash \vec{A} \to \mathtt{N}$. By the induction hypothesis, $\mathtt{cond}x.(f[\vec{d_i}], g[\vec{d_i}, x])m\vec{a'} : \mathtt{N}$ is not total, where $\vec{a_i} = m\vec{a'}$ and $m \in \mathbb{N}$. We first consider the subcase $m = 0$. Define e_{i+1} by the parent node whose term is $f[\vec{x}]$, and define $\vec{d_{i+1}} = \vec{d_i}$ and $\vec{a_{i+1}} = \vec{a'}$. We obtain (i), (ii), (iii) for $i+1$, as expected. We also have (iv) since the connection from $(\vec{d_i}; \vec{a_i}) = (\vec{d_i}; 0\vec{a'})$ to $(\vec{d_{i+1}}; \vec{a_{i+1}}) = (\vec{d_i}; \vec{a'})$ is trace compatible. Next we consider the subcase $m = \mathtt{S}(m')$. Define e_{i+1} by the parent node whose term is $g[\vec{x}, x]$, and define $\vec{d_{i+1}} = \vec{d_i}m'$ and $\vec{a_{i+1}} = \vec{a'}$. We obtain (i), (ii), (iii) for $i+1$, as expected. We also have (iv) since the connection from $(\vec{d_i}; \vec{a_i}) = (\vec{d_i}; \mathtt{S}(m')\vec{a'})$ to $(\vec{d_{i+1}}; \vec{a_{i+1}}) = (\vec{d_i}m'; \vec{a'})$ is trace compatible.

Hence, by the above construction, we have an infinite path $\vec{e} = (e_0, e_1, \ldots)$ in Π. Since Π satisfies the global trace condition, $\vec{e}$ contains a progressing trace $(k_m, k_{m+1}, \ldots)$, where, for each $m \le i$, k_i is an atom index of t_i, and k_{i+1}, t_{i+1} is the successor of k_i, t_i. Let n_i be an numeral in $\vec{d_i}, \vec{a_i}$ at index k_i for each $m \le i$. Then the sequence $(n_m, n_{m+1}, \ldots$ decreases at each progressing point. This means that it decreases infinitely many times since $(k_m, k_{m+1}, \ldots)$ has infinitely many progressing point. Finally we have a contradiction.

By this theorem, we have the following corollaries.

Corollary 8.4 *$t : \Gamma \vdash A$ implies t is total.*

Corollary 8.5 *For any closed $t : \mathtt{N}$, there is $n \in \mathbb{N}$ such that $t \mapsto_{\mathtt{safe}}{}^* n$.*

9 Uniqueness of normal form for closed terms of CT-λ of type N

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

By the weak normalization result proved in **S**8, we will deduce: for all for all closed terms t of CT-λ of type $\mathtt{N}$, there is some $n \in \mathtt{N}$ 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 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 $A = \mathtt{N}$. Then $t \sim_0 u$ if and only if for all normal $t', u' : \mathtt{N}$, 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 CT-λ we have $t \sim_0 t$. If $A = \mathtt{N}$ this means that the normal form of all closed terms of CT-λ of type $\mathtt{N}$ is unique, which is our goal.

We first define $t \sim u$ for *any* terms of 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 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 $t : \mathtt{N}$ is a closed term t of CT-λ and $n \in \mathtt{N}$.

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 $t[\vec{x}] : \vec{D} \to \mathtt{N}$. Assume $\vec{a} : \vec{A}, \vec{d} : \vec{D}$ and $\vec{b} : \vec{A}, \vec{e} : \vec{D}$ are vectors of closed terms of 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 8, all closed term t of CT-λ of type $\mathtt{N}$ have a value. If t is confluent, then the value is unique. For confluent closed term t of CT-λ of type $\mathtt{N}$, "safe" reduction preserves the value. Indeed, if $t \mapsto u$ then u is a closed term of CT-λ of type $\mathtt{N}$, 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 CT-λ of type $\mathtt{N}$, 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 9.1 *Assume t is any well-typed term such that $\neg(t \sim t)$. Then there is some infinite path $\pi = \{t_i | i \in \mathtt{N}\}$ of t with some infinite sequence $\sigma = \{\sigma_i | i \in \mathtt{N}\}$, 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.*

10 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 $t \notin X$ 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 10.1 (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 only made of redexes. Suppose a path in t is only made of redexes $r = (\lambda x : T.b)(u)$ or $r = (\mathtt{cond}x.(f, g))(t)$, where $t = 0$ 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, if τ progresses, then $r = (\mathtt{cond}x.(f, g))(t)$ and τ to the type of the variable x bound by $\mathtt{cond}x.(f, g)$. The only way to obtain this is that $t = x$ and r is obtained by an η-rule, and in this case r is no redex, because $x \neq 0, \mathtt{S}(u)$.

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 10.2 *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.

11 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 11.1 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 $i \in \mathtt{N}$, 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 $k \in \mathtt{N}$, 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 $k \in \mathtt{N}$, $k > 0$, 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.

12 Equivalence between cyclic and non-cyclic system T

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

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

The opposite inclusion, from $\mathtt{CT}\text{-}\lambda$ to $\mathbb{T}$, it has been proved by proof-theory for the combinatorial version of circular $\mathbb{T}$. For $\mathtt{CT}\text{-}\lambda$, we will define instead an algorithm taking an infinite term in $\mathtt{CT}\text{-}\lambda$, 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 $\mathtt{N} \to A$. Each bud b defines a partial bijection between the occurrence of $\mathtt{N}$ in its context and type $\Gamma \vdash \mathtt{N} \to A$, and the occurrences of $\mathtt{N}$ in the context and type $\Gamma \vdash \mathtt{N} \to A$ 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 $\mathtt{N}$ inside the context and type $\Gamma \vdash \mathtt{N} \to A$ 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 $\mathtt{N}$ and number i. We will define a translation $t^\sigma \in \mathbb{T}$ of $t \in \mathtt{CT}\text{-}\lambda$.
5. All traces move from u to any of the occurrences of u inside u. Some traces of some $\mathtt{N}$ in $\Gamma \vdash \mathtt{N} \to A$ disappear, some other are moved to some other $\mathtt{N}$, 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 $x_{i_1} = x_i$, $x_{i_2} = x_j, \ldots, x_{i_k}$ for some $k \geq 1$. We extend $x_{i_1}, \ldots, x_{i_k}$ to $x_{i_1}, \ldots, x_{i_n}$ in a random way: we defined in this way a permutation σ on $\{1, \ldots, m\}$ by $\sigma(j) = i_j$ for $j \in \{1, \ldots, m\}$ 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 $\texttt{exch}_\tau(f)$. The term $\texttt{exch}_\tau \in \mathbb{T}$ 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 $t \in \texttt{CT-}\lambda$ we started from.