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

Stefano Berardi

Abstract

We explore the possibility of defining a circular syntax directly for λ-abstraction and simple types, instead of interpreting λ-abstraction through combinators first, then inserting a circular syntax afterwards. We introduce our circular syntax as a fixed point operator defined by cases.

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

Our motivation is that terms of CT-λ are much shorter than the equivalent terms written in $\mathbb{T}$. Beside, the circular syntax of CT-λ, using binders instead of combinators, could be more familiar for researchers working in the field of Type Theory.

1 Introduction

We will introduce $\mathbb{\Lambda}$, a set of infinite λ-terms with a circular syntax. The types of $\mathbb{\Lambda}$ are: the atomic type N, possibly type variables $\alpha, \beta, \ldots$, and all types $A \to B$ for any types A, B. Type variables are only used to provide examples and play a minor role in our paper.

The terms of $\mathbb{\Lambda}$ are possibly infinite trees representing expressions defined with 0,S,ap (application), variables x^T with a type superscript, λ (the binder for defining maps), and cond, a fixed point binder operator for the arithmetic conditional (i.e., for test on zero). If we have no type variables, the the trees in $\mathbb{\Lambda}$ represent partial functionals on N, provided we add reduction rules.

In this paper will consider the set of well-typed terms WTyped $\subseteq \mathbb{\Lambda}$, of terms with a unique type for the term and all its subtypes. We will consider GTC $\subseteq$ WTyped, the subset of well-typed circular λ-terms satisfying a condition called the global trace condition. Terms of GTC denote total functionals. We will consider Reg, the subset of terms which are regular trees (having finitely many subtrees). They are possibly infinite terms which are finitely presented, by a finite graph with possibly cycles. Eventually, we introduce the topic of this paper, the system CT-λ. 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 regulal 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. The circular syntax has the advange of writing much shorter terms. Besides, by introducing a circular syntax with binders, we hope to provide 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": fair reductions can reduce *inside* fixed points, but they never forget entirely the task of reducing *outside* all fixed points.

Eventually, 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 $\mathbb{\Lambda}$ of infinite circular terms, the subset WTyped of well-typed terms and a reduction relation for them.

Definition 2.1 (Types of $\mathbb{\Lambda}$)

1. *The types of $\mathbb{\Lambda}$ are: the type N of natural numbers, an infinite list $\alpha, \beta, \ldots$ of type variables, and with A, B also $A \to B$. We call them simple types, types for short.*
2. $\mathtt{Type}$ *is the set of simples types.*

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

Definition 2.2 (Terms of $\mathbb{\Lambda}$)

1. *The terms of $\mathbb{\Lambda}$ are all possibly infinite trees we can define with x^T (variable name with superscript $T \in \mathtt{Type}$), $\lambda x^T.t$ (binder, abstraction), $\mathtt{ap}(t,u)$, 0, $\mathtt{S}(t)$ (successor of t), $\mathtt{cond}x^N.(f,g)$ (binder and arithmetical conditional).*
2. *t is an immediate subterm of $\mathtt{S}(t)$, f, g are immediate subterm of $\mathtt{cond}x^N.(f,g)$ and t, u are immediate subterm of $t(u)$.*
3. *The subterm relation is the reflexive and transitive closure of the immediate subterm relation.*
4. *We write $\mathtt{SubT}(t)$ for the set of subterms of t.*
5. $\mathtt{Reg}$ *is the set of terms $t \in \mathbb{\Lambda}$ such that $\mathtt{SubT}(t)$ is finite. We call the terms of* $\mathtt{Reg}$ *the regular terms.*
6. *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 $(i+1) \leq n$.*
7. *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.*

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

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 $\mathbb{\Lambda}$ 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 $\mathbb{\Lambda}$. Regular terms can be represented by the subterm relation restricted to $\mathtt{SubT}(t)$: this relation defines a graph with possibly cicles.

An example of regular term: the term $t = \mathtt{cond}x^N.(0,t)$. 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 an infinite term, we first define the context for any term of $\mathbb{\Lambda}$.

Definition 2.3 (Types and Contexts of $\mathbb{\Lambda}$)

1. *A context of $\mathbb{\Lambda}$ is any finite list $\Gamma = ({x_1}^{A_1} : A_1, \ldots, {x_n}^{A_n} : A_n)$ of pairwise distinct variables, each assigned to its $A_1, \ldots, A_n \in \mathtt{Type}$.*
2. *We write $\mathrm{FV}(\Gamma) = \{{x_1}^{A_1}, \ldots, {x_n}^{A_n}\}$. We say that Γ is a context for $t \in \mathbb{\Lambda}$ if $\mathrm{FV}(t) \subseteq \mathrm{FV}(\Gamma)$.*
3. *If $\Gamma = ({x_1}^{A_1} : A_1, \ldots, {x_n}^{A_n} : A_n)$, $\Gamma' = (x'_1 : A'_1, \ldots, x'_n : A'_{n'})$ are context of $\mathbb{\Lambda}$, then we write $\Gamma \sqsubseteq \Gamma'$ if Γ is a sub-list of Γ', that is, if there is a (unique) increasing map $f : \{1, \ldots, n\} \to \{1, \ldots, n'\}$ such that $x_i = x'_{f(i)}$ and $A_i = A'_{f(i)}$ for all $i \in \{1, \ldots, n\}$.*
4. *If Γ is a context of $\mathbb{\Lambda}$, then $\Gamma \setminus \{x^T : T\}$ is the context obtained by removing $x_i^{A_i} : A_i$ from Γ if $x_i^{A_i} = x^T$. If $x \notin \mathrm{FV}(\Gamma)$ then $\Gamma \setminus \{x^T : T\} = \Gamma$.*

From the context for a term we can define a context for each subterm of the term. A context is a list of type assignment to variables: our variables already have a type superscript, so a type assignment is redundant for variables (not for terms). Yet, we add an assignment relation $t : T$ for uniformity with the notations in use in Type Theory.

Definition 2.4 (Inherited Contexts of $\mathbb{\Lambda}$) *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 $t = \lambda x^T.u, \mathtt{cond}x^N.(f,g)$ has inherited context Γ', Δ, then u and g have context $(\Gamma', \Delta) \setminus \{x^T : T\}, x^T$, 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: iff* $t = \mathtt{S}(u), f(a)$ *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 $\mathbb{\Lambda}$ and the subset `WTyped` of well-typed terms. The typing rules are the usual ones but for the conditional binder `cond`, and for a fresh rule η-rule for typing the application $t(x)$ of a term to a variable x of type 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 `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 `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 2.5 (Typing rules of $\mathbb{\Lambda}$) *Assume* $\Gamma = x_1{}^{A_1} : A_1, \ldots, x_n^{A_n} : A_1$ *is a context.*

1. `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* $\Gamma \vdash f : N \to T$ *then* $\Gamma, x^N : N \vdash f(x) : T$*.*
3. `var`*-rule. If* $x^A \in \Gamma$ *then* $\Gamma \vdash x^A$*.*
4. `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.* $\Gamma \vdash 0 : N$
7. `S`*-rule. If* $\Gamma \vdash t : N$ *then* $\Gamma \vdash \mathtt{S}(t) : N$*.*
8. `cond`*-rule. If* $\Gamma \vdash f : T$ *and* $\Gamma, x^T \vdash g : T$ *then* $\Gamma \vdash \mathtt{cond}x^N.(f, g) : N \to T$*.*

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 2.6 (Well-typed term of $\mathbb{\Lambda}$) *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* $B = \Pi(\pi)$*,* $B = \Pi(\pi \star u_1)$*, ...,* $B = \Pi(\pi \star u_n)$*. 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* `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. $t \in \mathbb{\Lambda}$ *is well-typed if* $\Gamma \vdash t : A$ *for some* Γ, A*.*
3. `WTyped` *is the set of well-typed* $t \in \mathbb{\Lambda}$*.*

1. Some term in $\mathbb{\Lambda}$ has no type, like $0(0)$. Some terms have more than one type. For instance if $t = \mathtt{cond}x^N.(t, t)(0)$ we can prove $x^N : N \vdash t : A$ 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^N with some $\Gamma \vdash a : N$ which is not a variable, then we replace η-rule with $\mathtt{ap}$-rule.

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

Our goal is to provide a set of well-formed term for $\mathbb{\Lambda}$ 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 $\mathbb{\Lambda}$.

Definition 2.7 (reduction rules for $\mathbb{\Lambda}$)

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

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

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

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

The reason for forbiding reductions inside $\mathtt{cond}x^N.(f, g)$ is that thorugh $\mathtt{cond}$ we will-express fixed-point equations. Therefore reductions for $\mathtt{cond}$ are in fact unfolding of the definition of a fixed point. Therefore reductions for $\mathtt{cond}$ can can easily loop, and they are "unsafe". For this reason, we first consider the minimum possible of reductions $\mathtt{cond}x^N.(f, g)(0) \mapsto_{\mathtt{cond}} f$ and $\mathtt{cond}x^N.(f, g)(\mathtt{S}(t)) \mapsto_{\mathtt{cond}} g[t/x]$: those on maximal $\mathtt{cond}$-expression. We consider no reduction inside the arguments f, g of $\mathtt{cond}$. 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 $v = \mathtt{cond}x^N.(0, (\lambda x.v(x)))$, then $v(n) : N$ has infinitely many β-redexes inside $\mathtt{cond}$, therefore infinitely many unsafe reductions are possible. There are only finitely many safe reduction from $v(n)$ instead: $v(n)$ $\mathtt{cond}$-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 $\mathtt{cond}$-step and one β-step and so forth. After n $\mathtt{cond}$-reductions and n β-reductions we get $v(0)$. With one last $\mathtt{cond}$-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 $\mathtt{cond}$-reduce $\lambda x.v(x)$ before assigning either 0 or $\mathtt{S}(u)$ 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 connection 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 $t : \{x_1{}^{A_1}, x_2{}^{A_2}\} \vdash B_3 \to N$. 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 $t' = \lambda x_1{}^{A_1}, x_2{}^{A_2}.t : (A_1, A_2, B_3 \to N)$, 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 $A_j = N$ or $B_j = N$ respectively.

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

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 $\vec{B} \to N$ 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 $C_j = N$.*

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

Two arguments of type N are in connection 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 connection.

3.1 Some connection examples

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

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

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

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

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

Remark that the first argument of f (colored in old gold) is in connection 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 connection with some argument higher in the proof.

Example 3.3 An example of atom connection for the rule `cond`.

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

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

3.2 A formal definition of atom connection

We define an injection

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

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

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

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 $k = \texttt{ins}(a, k)$.*
2. *if $t = f(a)$ and a is not a variable and $t' = f$ then $k' = \texttt{ins}(n, k)$. 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 connection.

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

$$\frac{{x_1}^N : \mathbf{N}, {x_2}^N : \mathbf{N} \vdash f : \mathbf{N}, \mathbf{N} \to N}{{x_1}^N : \mathbf{N}, {x_2}^N : \mathbf{N}, x^N : \mathbf{N} \vdash f(x) : \mathbf{N} \to N} \ (\eta\text{-rule})$$

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

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

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

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

The connection 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 $\mathbb{\Lambda}$) 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 connection 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 connection we define the global trace condition and terms CT-λ.

We say that a tree is regular if it has finitely many subtrees. $t = \mathtt{cond}x^N.(0,t)$ 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* $t_i = \mathtt{cond}x^N.(f,g)$ *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 $\mathtt{Sum}$ computing the sum on N, which is an infinite term defined by $\mathtt{Sum} = \lambda x^N.\mathtt{cond} z.(x, \mathtt{S}(\mathtt{Sum}(x)(z)))$. 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 (†) above to the (†) below. The only infinite path contains a progressing trace, a sequence of N's colored in **old gold**.

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

5.2 The Iterator

A second example. We define a term $\mathtt{It}$ of 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 $\mathtt{It}(f, a, n) = f^n(a)$ 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 $\mathtt{It} = \lambda f, a, x.I$ with $I = (\mathtt{cond} x^N.(a, f(I)))(x)$.

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

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

for all $n, m \in \mathtt{Num}$. 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 $\lambda x.\mathtt{cond} y.(\ [w],\ [w]@v(\mathtt{S}(x), y))\ )$ 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 $w = \mathtt{It}(f, a, x) : N$, for which we already checked the global trace condition, or in the context of $v(\mathtt{S}(x), y)) : N$.

Then the mark moves from the named argument y of $v(\mathtt{S}(x), y) : N$ to the unique unnamed argument N of $v(\mathtt{S}(x))$, and eventually to the second argument of $v : N, N \to N$. 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 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 $k \in N$. 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 β, cond-redexes at the same time. Also the normal form is not regular: it contains all terms $f^k(I)$ for $k \in N$, hence infinitely many pairwise different terms. These infinite sub-terms are of a particulary simple form, though. They are obtained by the repeating k times the assignment $z := f(z)$, then applying $z := I$ once to the result.

Apparently, I is some term of 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 I are not regular.

6 Subject reduction for terms of CT-λ of type N

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

7 Weak normalization for closed terms of CT-λ of type N

In this section we prove that every closed term of CT-λ (that is, well-typed, regular and with the global trace condition) normalizes with finitely many "safe" steps to some numeral $n \in N$. 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, we 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 the set of all terms of the form $\mathtt{S}^n(0)$ for some $n \in N$.

We define closed total terms as in Tait's normalization proof, and we define values. Values are always closed total terms, the only different between values and closed total terms is that if a value has type N then it is a numeral.

Here is the informal definition. We define values of type N, which are numerals (hence closed term). We define total terms t of type N, which are the closed terms of type N evaluating in at least one reduction path to a numeral, and include values of type N. Closed total terms and values of type $t : \alpha$ (with α type variable) coincide, both are all closed terms. Closed total terms and values of type $t : A \to B$ coincide, both are the terms mapping values to total terms. An open term is total if all substitutions of free variables with values produce a closed *total* term.

Below is the formal definition of values, closed total terms and total terms.

Definition 7.1 (Values and total term) *We define values and closed total terms on values by induction on types.*

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

We can assign values to all arguments of a term and to any sub-term chain.

Definition 7.2 (trace-compatible assignment) *Assume $t \in$ WTyped is a well-typed term of $\mathbb{\Lambda}$ in the context $\Gamma = \vec{x} : \vec{B}$. Assume $t[\vec{x}] : \vec{A} \to N$ has argument types $\vec{B}, \vec{A}$ in Γ. Assume $\pi = (t_1, \ldots, t_n) \in \mathtt{SubT}(t)$ is any chain of immediate subterms of t.*

1. *A value assignment $\vec{v}$ for t in Γ is any vector $\vec{v} = \vec{u}, \vec{a} : \vec{B}\vec{A}$ of closed values.*
2. *A trace-compatible assignment $\vec{v} = (\vec{v_1}, \ldots, \vec{v_n})$ for π is any vector of value assigments, each $\vec{v_i}$ value assignment for t_i, which satisfies the following condition. For all j argument of t_i assigned to $\mathtt{S}^a(0)$, all k argument of t_{i+1} assigned to $\mathtt{S}^b(0)$, if j corresponds to k then:*
   (1) if j progresses to k we have $a = b + 1$
   (2) if j does not progress to k we have $a = b$.

If an infinite path has a trace-compatible assignment, then all traces of the path progress only finitely many times.

Proposition 7.3 (Trace assigment) *If $t \in \mathtt{WTyped}$, $\pi \in \mathtt{Tree}(t)$ is an infinite path, ρ is a value assignment to π.*

1. *If an argument j of some $t_i \in \pi$ has type N and value $\mathtt{S}^n(0)$, then a trace from j progresses at most n times.*
2. *$t \notin \mathtt{GTC}$.*

Proof 1. By definition of trace-compatible assignment, whenever the trace progresses, the natural number n decreases by 1, and whenever the trace does not progress, the the natural number n remains the same. Thus, a trace from j progresses at most n times, as we wished to show.

2. By point 1. above, no trace from any argument in any term of the branch π of $\mathtt{Tree}(t)$ progresses infinitely many times. By definition of $\mathtt{GTC}$, we conclude that $t \notin \mathtt{GTC}$.

Closed total terms are closed by safe reductions and by application.

Lemma 7.4 *1. Let $t : A$ be a closed term and $t \mapsto_{\mathtt{safe}} u$. If u is total, then so is t.*
2. Let $f : A \to B$, $a : A$ be closed total terms. Then so is $f(a)$.
3. Let $t[\vec{x}] : \vec{A} \to N$ 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 all $t[\vec{u}]\vec{a} : N$ are total, then $t[\vec{x}]$ is total.

Proof 1. *Point 1.* is shown by induction on A. By the subject reduction property, u has type A.
   (1) We first show the first *base case*, namely when $A = N$. By the assumption, u is closed total. By definition of u closed total, we have $t \mapsto_{\mathtt{safe}} u \mapsto_{\mathtt{safe}}{}^* n$ for some $n \in \mathtt{Num}$. Hence $t : N$ is total.
   (2) We first show the second *base case*, namely when $A = \alpha$. Both t and u are closed terms of type α, therefore are closed total terms.
   (3) We show the *induction case*, namely when $A = (A_1 \to A_2)$. 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 that u is total. Hence by the induction hypothesis $t(a)$ is total. We obtain that $t : A_1 \to A_2$ is total.

2. *Point 2.* is shown by case reasoning. Assume that$f : A \to B$, $a : A$ are closed total terms, in order to prove that $f(a)$ is closed total .
   Assume A is a variable type or an arrow type. Then $a : A$ is a value because values and closed total terms of type A coincide, therefore $f(a) : A$ is closed total by definition of closed total.
   Assume $A = N$. Then $a \mapsto_{\mathtt{safe}} n$ for some $n \in \mathtt{Num}$ by definition of closed total. $f(n)$ is closed total by definition of closed total. $f(a)$ is closed total by $f(a) \mapsto_{\mathtt{safe}} f(n)$ and point 2. above.

3. *Point 3.* 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 $t[\vec{u}]a_0\vec{a'} : N$ is closed total for all values $\vec{a'}$. Then $t[\vec{u}]a_0 : \vec{A'} \to N$ is total by the induction hypothesis on $\vec{A'} \to N$. By definition $t[\vec{u}] : \vec{A} \to N$ is closed total, and so by definition $t[\vec{x}]$ is total, as we expected.

Let Π be a proof of $\Gamma \vdash t : A$ and e be a node of Π, that is, a chain $(t_1, \ldots, t_n) \in \mathtt{Tree}(t)$ of immediate subterms of t with $t_1 = t$. We write $\Pi(e)$ for $\Gamma_n \vdash t_n : A_n$ with Γ_n inherited contex of t_n and A_n the type assigned by Π to t_n.

Theorem 7.5 *Assume* $\Pi : \Gamma \vdash t : A$. *If* t *is not total, then* $t \notin \mathtt{GTC}$, *i.e.: there is some infinite path in* Π *with no infinite progressing trace.*

Proof Assume that t is not total and $t : \vec{x} : \vec{D} \vdash \vec{A} \to N$ has a proof Π in order to show that $t \notin \mathtt{GTC}$. By Proposition 7.3.2. it is enough to prove that Π has some infinite path π and some trace-compatible assignment ρ for π. By 3. of Lemma 7.4, there exist closed values $\vec{a} : \vec{A}$ and $\vec{d} : \vec{D}$ such that $t[\vec{d}]\vec{a}$ is not total. By induction on i, we construct a term e_i for each natural number i, and a value assignment $\vec{v_i} = (\vec{d_i}, \vec{a_i})$ for e_i, such that such that $\pi = (e_1, \ldots, e_n)$ is a chain of immediate subterms from $e_1 = t$ and $(\vec{v_1}, \ldots, \vec{v_n})$ is a trace-compatible assignment for π. We assume that:

(i) e_i is a node of Π, whose last term is $t_i : \vec{x_i} : \vec{D_i} \vdash \vec{A_i} \to N$ 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 assignment 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 $\mathtt{cond}x^N.(f, g)$ 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 $(e_1, \vec{d_1}, \vec{a_1})$ for the root node t of Π. We choose $e_1 = t$ and $(\vec{d}, \vec{a})$ values such that $t[\vec{d}](\vec{a})$ is not total. Points (i), (ii), (iii), (iv) are immediate.

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 on the last rule for the node e_{i+1} in Π.

1. The case of `weak`, namely $\Pi(e_i) = t[y_{f(1)}/x_1, \ldots, y_{f(n)}/x_n] : \Delta \vdash \vec{A_i} \to N$ is obtained from $\Gamma \vdash t : A$, 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 $x_i^{C_i} = y_{f(i)}^{B_{f(i)}}$ for all $1 \leq i \leq n$. By the induction hypothesis and (b), the conclusion $t[y_{f(1)}/x_1, \ldots, y_{f(n)}/x_n][d_{i,f(1)}/y_{f(1)}, \ldots, d_{i,f(n)}/y_{f(n)}]\vec{a_i} : N$ of `weak` 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 we 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.
2. The case of `var`-rule, namely $\Pi(e_i) = \Gamma \vdash x_j : C_j$, cannot be, because $t_i = [x_j/d_j] = d_j$ is closed total by assumption on $\vec{d}$.
3. The case of 0-rule, namely $\Pi(e_i) = \Gamma \vdash 0 : N$, cannot be, because $t_i = 0$ is total because it is a numeral.
4. The case of `S`-rule, namely $\Pi(e_i) = \Gamma \vdash \mathtt{S}(t_{i+1}) : N$ is obtained from $\Gamma \vdash t_{i+1} : N$. $\vec{a_i}$ is empty, and by the induction hypothesis $\mathtt{S}(t_{i+1}[\vec{d_i}]) : N$ is not closed total. Then, by the definition of closed total, $t_{i+1}[\vec{d_i}]$ is not closed 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 the empty list, respectively. We obtain (i), (ii), (iii), and (iv) for $i + 1$, as expected.
5. The case of `ap`-rule, namely $\Pi(e_i) = \Gamma \vdash t[\vec{x}](u[\vec{x}]) : \vec{A} \to N$ is obtained from $\Gamma \vdash B \to t[\vec{x}] : \vec{A} \to N$ and $\Gamma \vdash u[\vec{x}] : B$. By the induction hypothesis, $t[\vec{d_i}](u[\vec{d_i}])\vec{a_i} : N$ is not total. We argue by case on the statement: $u[\vec{d_i}] : B$ *is closed total.*

   (1) We first consider the subcase that $u[\vec{d_i}] : B$ *is closed total.* We define $a' : B$ by $a' = n \in \mathtt{Num}$ such that $u[\vec{d_i}] \mapsto_{\mathtt{safe}}{}^* n$ if $B = N$, by $a' = u[\vec{d_i}]$ otherwise. By lemma **??.??**, a' is a value. Then define $e_{i+1} = t[\vec{x}]$, and define $\vec{d_{i+1}} = \vec{d_i}$ and $\vec{a_{i+1}} = a', \vec{a_i}$. Using Lemma 7.4, 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: all connected arguments of type N of e_i and e_{i+1} are the same, because the only fresh argument of e_{i+1} is a' and no argument of e_i is connected to it.

(2) Next we consider the subcase that $u[\vec{d_i}] : B$ *is not closed total*. By lemma **??.??** there is a sequence of values $\vec{a'}$ such that $u[\vec{d_i}]\vec{a'} : N$ is not total. Define $e_{i+1} = u[\vec{x}]$, 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: all connected arguments of type N of e_i and e_{i+1} are in $\vec{d_i}$ and therefore are the same.

6. The case of η-rule, namely $\Pi(e_i) = f[\vec{x}](x) : \Gamma, x^N \vdash \vec{A} \to N$ is obtained from $f[\vec{x}] : \Gamma \vdash N \to \vec{A} \to N$, where $(x^N : N) \notin \Gamma$. By the induction hypothesis, $f[\vec{d'}]d\vec{a_i} : N$ is not total, where $\vec{d_i} = \vec{d'}, d$ and d is the value of the argument type of x^N. 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: all connected argument in $\vec{d_i}, \vec{a_i}$ and $\vec{d_{i+1}}, \vec{a_{i+1}}$ are the same. The only difference between the two assignments is that the value d of the last argument type N is moved to the value d of the first unnamed argument of f.
7. The case of λ-rule, namely $\Pi(e_i) = \lambda x^A.(t_{i+1}[\vec{x}, x]) : \Gamma \vdash A, \vec{A} \to N$ is obtained from $t_{i+1}[\vec{x}, x^A] : \Gamma, x^A \vdash \vec{A} \to N$. By the induction hypothesis, $(\lambda x.(t_{i+1}[\vec{d_i}, x]))a\vec{a'} : N$ is not total, where $\vec{a_i} = a\vec{a'}$. Then, by Lemma 7.4, $t_{i+1}[\vec{d_i}, a]\vec{a'}$ 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: all connected argument in $\vec{d_i}, \vec{a_i}$ and $\vec{d_{i+1}}, \vec{a_{i+1}}$ are the same. The only difference between the two assignments is that the value d of the first unnamed argument of t_i is moved to the value d of the last argument type N. This is the opposite of the movement we have in the η-rule.
8. The case of `cond`-rule, namely $\Pi(e_i) = \texttt{cond}x^N.(f[\vec{x}], g[\vec{x}, x]) : \Gamma \vdash C \to \vec{A} \to N$ is obtained from $f[\vec{x}] : \Gamma \vdash \vec{A} \to N$ and $g[\vec{x}, x] : \Gamma, x^N \vdash \vec{A} \to N$. By the induction hypothesis, $\texttt{cond}x^N.(f[\vec{d_i}], g[\vec{d_i}, x])m\vec{a'} : N$ is not total, where $\vec{a_i} = m\vec{a'}$ and $m \in \mathbb{N}$. We argue by cases on m.
   (1) 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: each argument of e_i is connected to some equal argument of e_{i+1}, the first argument of e_i disappears but it is connected to no argument in f.
   (2) 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: each argument of e_i is connected to some equal argument of e_{i+1}, but for the argument m of e_i which is connected to m' in the last argument type N. This is fine because in the second premise of a `cond` the trace progress. This requires that *the numeral m decreases by* 1, as indeed it is the case.

Hence, by the above construction, we have an infinite path $\pi = (e_1, e_2, \ldots)$ in Π and a trace-compatible assignment, as we wished to show.

From this theorem we derive the weak normalization result: every closed term of type N reduces to some numeral for at least one reduction path.

Corollary 7.6 *Assume* $t : \Gamma \vdash A$,

1. $t \in \mathtt{GTC}$ *implies that* t *is total.*
2. *For any closed* $t : N$, *there is numeral* $n \in \mathtt{Num}$ *such that* $t \mapsto_{\mathtt{safe}}{}^* n$.

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

Incomplete section: uncomment the input command (below in the source code) to make this section visible

9 Infinite reductions

Incomplete section: uncomment the input command (below in the source code) to make this section visible

10 Normalization and Fairness

Incomplete section: uncomment the input command (below in the source code) to make this section visible

11 Equivalence between cyclic and non-cyclic system T

Incomplete section: uncomment the input command (below in the source code) to make this section visible