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 $\mathbb{A}$, a set of infinite λ-terms with a circular syntax. The types of $\mathbb{A}$ 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 they play a minor role in our paper.

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

In this paper will consider two sets of terms:

1. the set `CT`-λ of well-typed terms in a circular syntax, which are equivalent to the set of terms in Gödel system $\mathbb{T}$.
2. The set of terms `GTC`, satisfying a condition called global trace condition, which are possibly non-recursive terms used to provide semantics for `CT`-λ

We introduce more sets of terms required as intermediate steps in the definition of `GTC` and `CT`-λ.

1. `WTyped` $\subseteq \mathbb{A}$, the set of well-typed terms, is the of terms having a unique type
2. `Reg` is the set of terms of $\mathbb{A}$ which are regular trees (i.e., having finitely many subtrees). They are possibly infinite terms which are finitely presented by a finite graph possibly having cycles.
3. `GTC` $\subseteq$ `WTyped` will be defined as the set of well-typed circular λ-terms satisfying the global trace condition and regular. Terms of `GTC` denote total functionals.

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 versions of $\mathbb{T}$, our system `CT`-λ uses binders instead of combinators. The circular syntax has the advange of writing much shorter terms while preserving decidability of termination. 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 in the right-hand side of any `cond` and Church-Rosser. We will prove normalization in the limit if we use reductions which are "fair": fair reductions can reduce *inside* the right-hand side of some `cond`, but they never forget entirely the task of reducing *outside* all such subterms.

Eventually, we will prove that the closed terms `CT`-λ (those without free variables) represent exactly the total computable functionals definable in Gödel system $\mathbb{T}$.

2 The set of infinite λ-terms

We define the set $\mathbb{A}$ of infinite circular terms, the subset `WTyped` of well-typed terms and a reduction relation for them.

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

1. *The types of $\mathbb{A}$ 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. `Type` *is the set of simples types.*
3. *We suppose be given a set* `var` *of all pairs x^T of a variable name x and a type $T \in$* `Type`*.*

We write $\mathbb{N}$ for the set of natural numbers: all $n \in \mathbb{N}$ are represented in $\mathbb{A}$ by some expression of type N which we call a *numeral*.

We code (countable) trees on a set I as any set T of non-empty lists $(i_1, \ldots, i_n)$ of elements of I closed by non-empty prefix and having a minimum (i), which we call the root of T. As usual, a leaf of T is a maximal element of T. A subtree $U \subseteq T$ is the set of nodes of T extending some $l \in T$. U is coded by l in T. We write $\texttt{nil} = ()$ for the empty list and $(i_1, \ldots, i_n)\star i$ for $(i_1, \ldots, i_n, i)$. Any node $m \in T$ has the form $m = l\star i$ for some list l, with $l \in T$ if m is not the root and $l = \texttt{nil}$ if m is the root.

Terms of $\mathbb{A}$ are trees.

Definition 2.2 (Terms of $\mathbb{A}$) *The terms of $\mathbb{A}$ are all possibly infinite trees we can define with $x^T \in$* `var` *(leaf), $\lambda x^T.t$ (node with child t),* $\texttt{ap}(t, u)$ *(node with children t, u),* 0 *(leaf),* $\texttt{S}(t)$ *(node with child t),* $\texttt{cond}(f, g)$ *(node with children f, g).*

More formally, a term is any set of finite non-empty lists on the set $I = \{x^T, \lambda x^T | x^T \in \texttt{var}\} \cup \{0, \texttt{S}, \texttt{ap}, \texttt{cond}\}$ closed by non-empty prefix, having a minimum, and whose maximal elements are all of the form x^T or 0.

We define the sub-term relation.

Definition 2.3 (Sub-terms of $\mathbb{A}$)

1. *A subterm of t is any subtree of t, coded by some node of t.*
2. *We call the child t of a node u any immediate sub-term of u.*
3. *We write* $\texttt{SubT}(t)$ *for the (finite or infinite) set of subterms of t. They are coded by the nodes of t.*
4. `Reg` *is the set of terms $t \in \mathbb{A}$ such that* $\texttt{SubT}(t)$ *is finite. We call the terms of* `Reg` *the regular terms.*
5. *We write* $\texttt{Tree}(t)$ *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 $(i + 1) \leq n$.*
6. *When $t = \texttt{S}^n(0)$ for some natural number $n \in \mathbb{N}$ we say that t is a numeral. We write* `Num` *for the set of numeral.*

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

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

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

An example of regular term: the term $t = \texttt{cond}(0, t)$. The set $\texttt{SubT}(t) = \{t, 0\}$ of subterms of t is finite, therefore t is a regular term. However, $\texttt{Tree}(t)$ is an infinite tree (it includes itself as a subtree). The finite branches of $\texttt{Tree}(t)$ are all $(t, t, t, \ldots, 0)$, the unique infinite branch of $\texttt{Tree}(t)$ is $(t, t, t, \ldots)$.

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

Definition 2.4 (Contexts of $\mathbb{A}$)

1. *A context of $\mathbb{A}$ 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 $A_1, \ldots, A_n \in$ `Type`.*
2. *We denote the empty context with* `nil`*. We write* `Ctxt` *for the set of all contexts.*
3. *A sequent is the pair of a context Γ and a type A, which we write as $\Gamma \vdash A$. We write* `Seq` $=$ `Ctxt` $\times$ `Type` *for the set of all sequents.*
4. *A typing statement is the list of a context Γ, a term t and a type A, which we write as $\Gamma \vdash t : A$. We write* `Stat` $=$ `Ctxt` $\times \mathbb{A} \times$ `Type` *for the set of all typing statements.*
5. *We write $\mathrm{FV}(\Gamma) = \{{x_1}^{A_1}, \ldots, {x_n}^{A_n}\}$. We say that Γ is a context for $t \in \mathbb{A}$ and we write $\Gamma \vdash t$ if $\mathrm{FV}(t) \subseteq \mathrm{FV}(\Gamma)$.*
6. *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 $\mathbb{A}$, then we write*
   (1) $\Gamma \subsetneqq \Gamma'$ if for all $(x^A : A)$: if $(x^A : A) \in \Gamma$ then $(x^A : A)\Gamma'$
   (2) $\Gamma \sim \Gamma'$ if for all $(x^A : A)$: $(x^A : A) \in \Gamma$ if and only if $(x^A : A)\Gamma'$
7. *If Γ is a context of $\mathbb{A}$, 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 $\Gamma \setminus \{x^T : T\} = \Gamma$.*

We have $\Gamma \subsetneqq \Gamma'$ if and only if if there is a (unique) map $\phi : \{1, \ldots, n\} \to \{1, \ldots, n'\}$ such that $x_i = x'_{\phi(i)}$ and $A_i = A'_{\phi(i)}$ for all $i \in \{1, \ldots, n\}$. We have $\Gamma \sim \Gamma'$ if and only if $\Gamma \subsetneqq \Gamma'$ and $\Gamma \supsetneqq \Gamma'$ if and only if Γ, Γ' are permutation each other if and only if the map ϕ above is a bijection.

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 our variables (not for our terms). Yet, we add an assignment relation $x^T : T$ for uniformity with the notation $x : T$ in use in Type Theory.

We define typing rules for terms of $\mathbb{A}$ and the subset `WTyped` of well-typed terms. We consider a term well-typed if typing exists and it is unique for the term and for all its subterms.

The typing rules are the usual ones but for a fresh rule `apvar` for typing the application $t(x^T)$ of a term to a variable x^T. `apvar` is but the particular case of `ap` rule when the argument is a variable of the context. We prefer to use a separate name for `apvar` because of the special role that `apvar` has in this paper for making easier to decide the termination of rewriting. We reserve the name `ap` for the typing rule of an application $t(u)$ of a term to a non-variable u.

Remark that we introduced a unique *term notation* for application: we write $\mathtt{ap}(t, u)$ no matter if u is a variable or not.

We have a a single structural rule `weak` for extending a context Γ to a context $\Gamma' \supsetneqq \Gamma$. When $\Gamma' \sim \Gamma$ (Γ', Γ are permutation each other), the rule `weak` can be used for variable permutation. Variable renaming for a term $t[\vec{x}]$ can be obtained by writing $(\lambda\vec{x}.t[\vec{x}])(\vec{x'})$. Therefore we do not assume having a primitive rule for it.

Definition 2.5 (Typing rules of $\mathbb{A}$) *Assume $\Gamma = {x_1}^{A_1} : A_1, \ldots, x_n^{A_n} : A_1$ is a context.*

1. *A rule is a list of $n+1$ typing statements: $\Gamma \vdash t_1 : A_1, \ldots, \Gamma \vdash t_n : A_n, \Gamma \vdash t : A$. The first n sequents are the premises of the rule, the last sequent is the conclusion of the rule.*
2. *We read the rule above: "if $\Gamma \vdash t_1 : A_1, \ldots, \Gamma \vdash t_n : A_n$ then $\Gamma \vdash t : A$".*

We list the typing rules of $\mathbb{A}$.

1. `weak`*-rule (Weakening). If $\Gamma \vdash t : T$ and $\Gamma \subsetneqq \Gamma'$ then $\Gamma' \vdash t : T$*
2. `var`*-rule. If $x^A \in \Gamma$ then $\Gamma \vdash x^A : A$.*
3. `ap`*-rule. If $\Gamma \vdash f : A \to B$ and $\Gamma \vdash a : A$ and a is not a variable then $\Gamma \vdash f(a) : B$.*
4. `apvar`*. If $\Gamma \vdash f : A \to B$ and $(x^A : A) \in \Gamma$ then $\Gamma \vdash f(x^A) : B$.*
5. *λ-rule. If $\Gamma, x^A : A \vdash b : B$ 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 $\Gamma \vdash g : N \to T$ then $\Gamma \vdash \mathtt{cond}(f, g) : N \to T$.*

We abbreviate $\mathtt{nil} \vdash t : A$ *($t : A$ in the empty context) with* $\vdash t : A$*. We write the set of rules of* $\mathbb{A}$ *as*

$$\mathtt{Rule} = \{r \in \mathtt{Seq} \cup \mathtt{Seq} \cup \mathtt{Seq}^2 | r \textit{ instance of some rule of } \mathbb{A}\}$$

Proposition 2.6 (Rules and subterms) *Assume* $r, r' \in \mathtt{Rule}$ *have conclusion* $\Gamma \vdash t : A$, $\Gamma' \vdash t' : A'$ *respectively, and* $r, r \neq \mathtt{weak}$*.*

1. *If* $\Gamma \vdash t : A = \Gamma' \vdash' t' : A'$ *then* $r = r'$*.*
2. *If the premises of* r *are some* $\Gamma_1 \vdash t_1 : A_1, \ldots, \Gamma_n \vdash t_n : A_n$*, then the list of immediate subterms of* t *is exactly* $t_1, \ldots, t_n$*.*

From the typing rules we define the proofs that a term is well-typed. Proofs are trees on the set $\mathtt{Rule}$, each node is associated to a rule. We forbid applying infinitely many consecutive weakening rules. The reason is that we can prove any typing statement $\Gamma \vdash t : A$ by infinitely many consecutive weakening rules.

Therefore eventually any branch includes a non-weakening rule after finitely many weakening rules. Any finite composition of weakening rules can be contracted to one weakening rule, therefore we can assume that weakening and non-weakening rules alternate. By Prop. 2.6 non-weakening rules are uniquely determined by their conclusion $\Gamma \vdash t : A$, and have assumptions $\Gamma_1 \vdash t_1 : A_1, \ldots, \Gamma_n \vdash t_n : A_n$, where $t_1, \ldots, t_n$ exactly the immediate subterms of t. Therefore we can further simplify the notion of proof to a decoration θ of the subterm tree by sequents. Any branch $\pi\star u$ decorated with $\theta(\pi\star u) = \Delta \vdash B$ represents a rule $r \neq \mathtt{weak}$ of conclusion $\Delta \vdash u : B$. The rule r is possibly followed by a single weakening inferring some assumption of the father node $\theta(\pi)$, and inferring the conclusion of the proof if $\pi = \mathtt{nil}$.

This is the formal definition of proof.

Definition 2.7 (Well-typed term of $\mathbb{A}$) *Assume* $\Pi : \mathtt{Tree}(t) \to \mathtt{Rule}$ *is an assignment of one non-weakening rule* $r = \Pi(\pi)$ *of conclusion some* $\Delta \vdash u : B$ *for each chain* π *of immediate subterms from* t *to* u*. Assume* Γ *is a context of* t *(i.e,* $\mathrm{FV}(t) \subseteq \Gamma$*) and* $A \in \mathtt{Type}$

Then we write $\Pi : \Gamma \vdash t : A$*, and we say that* Π *is a proof of* $\Gamma \vdash t : A$ *if:*

1. $\Pi((t))$ *is a rule of conclusion* $\Gamma' \vdash A$ *for some* $\Gamma' \subsetapprox \Gamma$*.*
2. *Assume that*
   (1) π *is any subterm chain from* t *to* u
   (2) $u_1, \ldots, u_m$ *are the immediate subterms of* u
   (3) $r_1 = \Pi(\pi\star u_1), \ldots, r_n = \Pi(\pi\star u_n$ *are rules of conclusion* $\Delta'_1 \vdash B_1, \ldots, \Delta'_n \vdash B_n$*.*
   (4) $r = \Pi(\pi)$ *is a rule of premises* $\Delta_1 \vdash B_1, \ldots, \Delta_n \vdash B_n$

   Then $\Delta'_1 \subsetapprox \Delta_1, \ldots, \Delta'_n \subsetapprox \Delta_n$*.*

Using all notations above, we require that the following weakening rules are correct:

$$\cfrac{\cfrac{\Delta'_1 \vdash u_1 : B_1}{\Delta_1 \vdash u_1 : B_1}\,\mathtt{weak} \quad \ldots \quad \cfrac{\Delta'_n \vdash u_1 : B_1}{\Delta_n \vdash u_1 : B_1}\,\mathtt{weak}}{\Delta \vdash u : B}\,r$$

and

$$\frac{\Gamma' \vdash t : A}{\Gamma \vdash t : A}\,\mathtt{weak}$$

Eventually we define the well-typed terms of $\mathbb{A}$.

Definition 2.8 (Well-typed term of $\mathbb{A}$)

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 and only if* $\Pi : \Gamma \vdash t : A$ *for some unique* A *and some context* $\Gamma \supsetapprox \mathrm{FV}(t)$*.*
3. $\mathtt{WTyped}$ *is the set of well-typed* $t \in \mathbb{A}$*.*

We provide some examples of well-typed and not well-typed terms. Some term in $\mathbb{A}$ has no type, like the application $0(0)$ of the non-function 0. Some term in $\mathbb{A}$ has more than one type. For instance if $t = \texttt{cond}(t,t)(0)$ we can prove $\vdash t : A$ for all types $A \in \texttt{Type}$. The subterms of t are $\{t, \texttt{cond}(t,t), 0\}$ and a proof $\Pi :\vdash t : A$ is defined by $\Pi(\pi\star t) =\vdash A$, $\Pi(\pi\star\texttt{cond}(t,t)) = N \to A$ and $\Pi(\pi\star 0) = N$. A term with two types has the leftmost branch infinite, as it is the case for t above. We will prove that if *all* subterms of a term have the leftmost branch finite, then the term has at most one type. If the term is also regular then we can decide if the typing proof exists and if it exists we can compute it (in quadratic time). Well-typed terms are closed by substitution.

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 2.9 (reduction rules for $\mathbb{A}$)

1. $\mapsto_\beta$: $(\lambda x^A.b)(a) \mapsto_\beta b[a/x]$
2. $\mapsto_{\texttt{cond}}$: $\texttt{cond}(f,g)(0) \mapsto_{\texttt{cond}} f$ *and* $\texttt{cond}(f,g)(\texttt{S}(t)) \mapsto_{\texttt{cond}} g(t)$.
3. $\mapsto$ *is the context and transitive closure of* $\mapsto_\beta$ *and* $\mapsto_{\texttt{cond}}$
4. $t \sim u$ *if and only if there is some* $v \in \mathbb{A}$ *such that* $t \mapsto v$ *and* $u \mapsto v$.
5. *The* `cond`*-depth of a subterm path* $(t_1, \dots, t_n) \in \texttt{SubT}(t)$ *is the number of* $i \in [1, n[$ *such that* $t_i = \texttt{cond}(f,g)$ *and* $t_{i+1} = g$ *(such that* t_n *occurs in the right-hand-side of a* `cond`*).*
6. *We say that* $t \mapsto_{\texttt{safe}} u$*, or that* t *reduces safely to* u*, if we only reduce in a subterm path of* `cond`*-depth* $= 0$.
7. *A term is safe-normal if all its redexes (if any) have* `cond`*-depth* > 0.

An example. Let $u = \texttt{cond}.(0, (\lambda z.u)(z))$, where we omitted the type superscript of z. Then u is safe-normal, because all redexes in u are of the form $(\lambda z.u)(z)$ and in the right-hand-side of a `cond`. However, the tree form of u has the following branch:

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

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

The reason for forbiding reductions in the right-hand-side $\texttt{cond}(f,g)$ is that through branches crossing the right-hand-side of some `cond` we will-express fixed-point equations. Therefore reductions on these branches can easily loop, and they are "unsafe". For this reason, we first consider the minimum possible of reductions of the form: $\texttt{cond}(f,g)(0) \mapsto_{\texttt{cond}} f$ and $\texttt{cond}(f,g)(\texttt{S}(t)) \mapsto_{\texttt{cond}} g[t/x]$: only those on maximal `cond`-expression. We consider no reduction inside the second argument of `cond`. In a second moment, by adding a restriction of *fairness* on reduction strategies, we will be able to recover strong normalization for most "unsafe" reductions.

We include one example of safe `cond`-reductions of a term $v(n)$ to a normal form. Assume $n \in \texttt{Num}$ is any numeral and $v = \texttt{cond}(0, v)$. There are only finitely many reductions from $v(n)$ instead, and they are all safe. $v(n)$ `cond`-reduces to $v(n-1)$, then we loop: $v(n-1)$ `cond`-reduces to $v(n-2)$ and so forth. After n `cond`-reductions we get $v(0)$. With one last `cond`-reduction we get 0 and we stop. Thus, the term $v(n)$ strongly normalizes in $(n+1)$-steps.

3 The trace of the cyclic λ-terms

We introduce the set `GTC` of the set `WTyped` of well-typed terms of $\mathbb{A}$ satisfying a condition call *global trace condition* for all proofs that the term is well-typed.

In order to define the global trace condition, first we define a notion of trace for possibly infinite λ-terms, describing how an input of type N is used when computing the 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 of *index of atomic types* for a term.

We sketch the notion of connection through an example. Assume ${x_1}^{A_1} : A_1, {x_2}^{A_2} : A_2 \vdash t[x_1, x_2] : B_3 \to N$. Then the list of argument types of t is A_1, A_2, B_3. A_1, A_2 are arguments with names x_1, x_2, and B_3 is an unnamed argument. Remark that for an open term $t[x_1, x_2]$ 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 argument type 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* $\vec{A} = A_1, \ldots, A_n$, $\vec{B} = B_{n+1}, \ldots, B_{n+m}$, $\Gamma = \{\vec{x} : \vec{A}\}$, *and* $\Gamma \vdash t : \vec{B} \to N$.

1. *The list of argument types of* t *is* $\vec{C} = \vec{A}, \vec{B}$.
2. $A_1, \ldots, A_n$ *are the named arguments, with names* $x_1, \ldots, x_n$.
3. $B_{n+1}, \ldots, B_{n+m}$ *are the unnamed arguments.*
4. *An index of an an argument type 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 $\mathbb{A}$. The definition of atom connection for a syntax including the binder λ is the main contribution of this paper. Before providing the general definition, we discuss the notion of connection through examples. We draw in the same color two arguments of type N which are in connection.

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

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

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.

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

Remark that the first unnamed 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 $f(a)$, does not receive a value from outside the term. However, the first argument of f can be in connection with some argument higher in the typing proof.

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

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

Remark that the first unnamed 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.

3.1 A formal definition of atom connection

We define an injection

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

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

Definition 3.2 (Atom connection 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* $\Gamma \vdash t : \vec{B} \to N$.

For each atom index k *in* t, *for* $t' = t$ *or* t' *immediate subterm 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* $t' = t$, *adding a variable in position* $p + 1$, *then* $k = \texttt{ins}(p, k')$.
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* $t = \texttt{cond}(f, g)$ *and* $t' = f$ *then* $k' = \texttt{ins}(n, k)$. *If* $t' = g$ *then* $k' = k$.

We require $k = k'$ in all other cases, which are: `S`*,* λ*,* `apvar`*.*

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 `apvar`. We assume that x is a variable.

$$\frac{x_1 : \mathbf{N}, x_2 : \mathbf{N} \vdash f : \mathbf{N}, \mathbf{N} \to N}{x_1 : \mathbf{N}, x_2 : \mathbf{N}, x : \mathbf{N} \vdash f(x) : \mathbf{N} \to N} \ (\texttt{apvar})$$

Remark that the first unnamed 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.

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

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

Summing up, if we move up in a `apvar`-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 connection on atoms in a proof $\Pi : \Gamma \vdash t : A$ defines a graph $\texttt{Graph}(\Pi)$ whose nodes are all pairs (k, u), with u subterm of t and k index of some atomic argument of u. $\texttt{Graph}(\Pi)$ is finite if t is regular. We define a trace as a (finite or infinite) path in the graph $\texttt{Graph}(\Pi)$.

Definition 3.3 (Trace for well-typed terms in $\mathbb{A}$) *Assume Π is any typing proof of $t \in$* `WTyped`*.*

1. *A path of t is any finite or infinite list $\pi = (t_0, \ldots, t_n, \ldots)$ such that for all t_{i+1} in π we have $t_{i+1} = t_i$ or $t_{i+1} = t_i$ immediate subterm of t_i.*
2. *Assume $\pi = (t_0, \ldots, t_n, \ldots)$ is a path of t, finite or infinite. A finite or infinite trace τ of π in Π is a list $\tau = ((k_m, t_m), \ldots, (k_n, t_n), \ldots)$ such that for all $i = m, \ldots, n, \ldots$:*
   (1) k_i is the index of a type N 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. $t = \texttt{cond}(0, t)$ is an infinite regular tree, with subtrees: t, 0.

Definition 4.1 (Global trace condition and terms of CT-λ) *Assume $\tau = ((k_m, t_m), \ldots, (k_n, t_n), \ldots)$ is a trace of a path $\pi = (t_1, \ldots, t_n, \ldots)$ of $t \in$* `WTyped`*. Assume $i = m, \ldots, n$.*

1. *τ is progressing in i if $t_i =$ `cond`(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 typing proofs Π, of t, all infinite paths π of t in Π, there is some infinitely progressing path τ in π and 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.*

The notion of global trace condition is defined through a universal quantification on proof $\Pi : \Gamma \vdash t : A$, and proofs Π are possibly infinite objects. We would expect that the global trace condition is not decidable. Surprisingly, it is decidable in polynomial space in t. The reason is that is some proof satisfies the global trace condition then all proofs do, and for regular terms profos are finite graphs.

Besides, the global trace condition should be decidable quite fast for realistic λ-terms.

5 Examples of terms of $\texttt{CT-}\lambda$

5.1 The sum map

A first example of term of $\texttt{CT-}\lambda$. In this example, the type superscript N of variables x^N, z^N is omitted. We provide a term $\texttt{Sum}$ computing the sum on N, which is an infinite term defined by $\texttt{Sum} = \lambda x.\texttt{cond}(x, \lambda z.\texttt{S}(\texttt{Sum}(x)(z)))$. This term is regular with subterms

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

This term is well-typed by the following circular derivation with a back edge from the $\dagger$ above to the $\dagger$ below. We colored in old gold one sequence of atoms N (one trace of the proof). This is a cyclic infinitely progressing trace in the unique infinite path of the proof. We marked $\spadesuit$ the only progress point.

$$\dfrac{\dfrac{\dfrac{}{x : N \vdash x}\,\texttt{var} \qquad \dfrac{\dfrac{\dfrac{\dfrac{\vdash \texttt{Sum} : N, N \to N \quad (\dagger)}{x : N, z : N \vdash \texttt{Sum} : N, N \to N}\,\texttt{weak}}{x : N, z : N \vdash \texttt{Sum}(x) : N \to N}\,\texttt{apvar}}{x : N, z : N \vdash \texttt{Sum}(x)(z) : N}\,\texttt{apvar}}{x : N, z : N \vdash \texttt{S}(\texttt{Sum}(x)(z)) : N \quad (\spadesuit)}\,\texttt{S}}{x : N \vdash \texttt{cond}(x, \lambda z.\texttt{S}(\texttt{Sum}(x)(z))) : N \to N}\,\texttt{cond}}{\vdash \texttt{Sum} : N, N \to N \quad (\dagger)}\,\lambda$$

5.2 The Iterator

A second example. We define a term $\texttt{It}$ of $\texttt{CT-}\lambda$ computing the iteration of maps on N. $\texttt{It}$ is a normal term of type $(N \to N), N, N \to N$ such that $\texttt{It}(f, a, n) = f^n(a)$ for all numeral $n \in \texttt{Num}$. We have to to solve the equations $\texttt{It}(f, a, 0) \sim a$ and $\texttt{It}(f, a, \texttt{S}(t)) \sim f(\texttt{It}(f, a, t))$ where f, a abbreviate $f^{N \to N}$ and a^N. We solve them with $\texttt{It} = \lambda f, a.\texttt{it}$ where $\texttt{it} = \texttt{cond}(a, \lambda x.f(\texttt{it}(x))) : N \to N$ is in the context $\Gamma = (f : N \to N, a : N)$.

The term is well-typed and regular by definition. We check the global trace condition. We follow the trace of the last unnamed argument N of $\texttt{It}$. The trace moves to $\lambda a.\texttt{it}$, then to $a : N$ in the context of $\texttt{it}$. Then the trace either moves to the name a and there the path stops, or the trace progresses and moves to the context of $f(\texttt{it}(x))$, then to the context of f or the context of $\texttt{it}(x)$. Either the path stops in f, or the path moves to $\texttt{it}(x)$ and then to $\texttt{it}$. Now we loop: if the path continues forever, then after infinitely many step the trace progresses infinitely many times, each time it moves to the right-hand side of a $\texttt{cond}$-rule. We sum up our discussion in the picture below. Again, we have a back edge from the $\dagger$ above to the $\dagger$ below and we marked $\spadesuit$ the only progress point. We abbreviate $\Gamma = (f : N \to N, a : N)$.

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma \vdash a : N}\,\texttt{var} \qquad \dfrac{\dfrac{\dfrac{}{\Gamma, x : N \vdash f : N}\,\texttt{var} \qquad \dfrac{\Gamma, x : N \vdash \texttt{it} : N \to N \quad (\dagger)}{\Gamma, x : N \vdash \texttt{it}(x) : N \to N}\,\texttt{apvar}}{\Gamma, x : N \vdash f(\texttt{it}(x)) : N}\,\texttt{ap}}{\Gamma \vdash \lambda x.f(\texttt{it}(x)) : N \to N}\,\lambda}{\Gamma \vdash \texttt{cond}(a, \lambda x.f(\texttt{it}(x))) : N \to N \quad (\spadesuit, \dagger)}\,\texttt{cond}}{f : N \to N \vdash \lambda a.\texttt{it} : N, N \to N}\,\lambda}{\vdash \texttt{It} : (N \to N), N, N \to N}\,\lambda$$

5.3 The Interval Map

A third example. We simulate lists with two variables $\texttt{nil} : \alpha$ and $\texttt{cons} : N, \alpha \to \alpha$. We recursively define a notation for lists by $[] = \texttt{nil}$, $a@l = \texttt{cons}(a, l)$ and $[a, \vec{a}] = a@[\vec{a}]$. We add no elimination rules for lists, though, only the variables $\texttt{nil}$ and $\texttt{cons}$. Elimination rules are not required in our example.

We will define a term $\texttt{In}$ with one argument $f : N \to N$ and three argument $a, x, y : N$ (we skip type superscripts), such that

$$\texttt{In}(f, a, n, m) \mapsto [f^n(a), f^{n+1}(a), \ldots, f^{n+m}(a)]$$

for all numeral $n, m \in \texttt{Num}$. We have to to solve the recursive equations

$$\texttt{In}(f, a, n, 0) \sim [f^n(a)] \qquad \texttt{In}(f, a, n, \texttt{S}(m)) \sim f^n(a)@\texttt{In}(f, a, \texttt{S}(n), m)$$

Assume $\texttt{it} = \texttt{cond}(a, \lambda x.f(\texttt{it}(x))) : N \to N$ is defined as in the previous sub-section: in particular, $\texttt{it}(n) \sim f^n(a)$. We solve the recursive equation for $\texttt{In}$ with $\texttt{In} = \lambda f, a.\texttt{in}$, where

$$f : N \to N, a : N \vdash \texttt{in} : N, N \to \alpha$$

is defined by

$$\texttt{in} = \lambda x.\texttt{cond}(\ [\texttt{it}(x)],\ \lambda y.\texttt{it}(x)@(\lambda x.\texttt{in})(\texttt{S}(x))(y)\ )$$

The term is well-typed and regular by definition. We check the global trace condition. Any infinite path either moves to $\texttt{it}$, for which we already checked the global trace condition, or cyclically moves from $\texttt{in}$ to $\texttt{in}$. We follow the trace of the last unnamed argument N of $\texttt{In}$ inside this infinite path. The trace moves to the last unnamed argument N of $\texttt{in} : N, N \to N$, then to the last unnamed argument of $\texttt{cond}(\ [\texttt{it}(x)],\ \lambda y.\texttt{it}(x)@(\lambda x.\texttt{in})(\texttt{S}(x))(y)\ )$. In this step the trace progresses, and moves to the first unnamed argument of $\lambda y.\texttt{it}(x)@(\lambda x.\texttt{in})(\texttt{S}(x))(y)\ )$, then to $y : N$ in the context of $\texttt{it}(x)@(\lambda x.\texttt{in})(\texttt{S}(x))(y)\ )$. After one $\texttt{apvar}$ rule, the trace reaches the unique unnamed argument of $(\lambda x.\texttt{in})(\texttt{S}(x))$, then the last unnamed argument of $\texttt{in}$. From $\texttt{in}$ we loop. Each time the trace moves from $\texttt{in}$ to $\texttt{in}$ then the trace progresses once. We conclude the global trace condition.

We have infinitely many nested β-reduction $(\lambda x. \ldots)(\texttt{S}(x))$. We can remove all of them in a single step. Inside the β-redex number k we obtain a sub-term $\texttt{it}[\texttt{S}(x)/x] \ldots [\texttt{S}(x)/x]$ (substitution repeated k times). The result is $\texttt{it}[\texttt{S}^k(x)/x]$. The nested substitution produce infinitely many pairwise different sub-terms $\texttt{it}(\texttt{S}^k(x))$ for all $k \in N$. We need infinitely many steps to normalize all $\texttt{it}(\texttt{S}^k(x))$ to $f^k(I)$, even if we allow to reduce all $\beta, \texttt{cond}$-redexes at the same time. Also the normal form is not regular: it contains all terms $f^k(\texttt{it}(x))$ for $k \in N$, hence infinitely many pairwise different terms.

Summing up, $\texttt{In}$ is some term of $\texttt{CT-}\lambda$ 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 $\texttt{In}$ are not regular.

6 Subject reduction for terms of $\texttt{CT-}\lambda$ of type N

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

7 Weak normalization for closed terms of $\texttt{CT-}\lambda$ of type N

In this section we prove that every closed term of $\texttt{CT-}\lambda$ (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 $\texttt{S}^n(0)$ 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 $t \mapsto_{\texttt{safe}}{}^* u$ 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 $t \in \texttt{WTyped}$ is a well-typed term of $\mathbb{A}$ 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 $\pi = (t_1, \dots, t_n) \in \texttt{SubT}(t)$ 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 $\vec{v} = (\vec{v_1}, \dots, \vec{v_n})$ 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 $\texttt{S}^a(0)$, all k argument of t_{i+1} assigned to $\texttt{S}^b(0)$, 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 $t \in \texttt{WTyped}$, $\pi \in \texttt{Tree}(t)$ 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 $\texttt{S}^n(0)$, then a trace from j progresses at most n times.*
2. *$t \notin \texttt{GTC}$.*

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 $\texttt{Tree}(t)$ progresses infinitely many times. By definition of $\texttt{GTC}$, we conclude that $t \notin \texttt{GTC}$.

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

Lemma 7.4 *1. Let $t : A$ be a closed term and $t \mapsto_{\texttt{safe}} u$. 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 $t \mapsto_{\texttt{safe}} u \mapsto_{\texttt{safe}}{}^* n$ for some $n \in \texttt{Num}$. 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 $t(a) \mapsto_{\texttt{safe}} u(a)$ 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 $(t_1, \dots, t_n) \in \mathtt{Tree}(t)$ 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 $\pi = (e_1, \dots, e_n)$ is a chain of immediate subterms from $e_1 = t$ and $(\vec{v_1}, \dots, \vec{v_n})$ 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 $\mathtt{cond}(f, g)$ 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 $\Pi(e_i) = t[y_{f(1)}/x_1, \dots, y_{f(n)}/x_n] : \Delta \vdash \vec{A_i} \to N$ is obtained from $\Gamma \vdash t : A$, where $\Gamma = x_1 : C_1, \dots, x_n : C_n$, $\Delta = y_1 : B_1, \dots, y_m : B_m$, $f : \{1, \dots, n\} \to \{1, \dots, m\}$ 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 $t[y_{f(1)}/x_1, \dots, y_{f(n)}/x_n][d_{i,f(1)}/y_{f(1)}, \dots, d_{i,f(n)}/y_{f(n)}]\vec{a_i} : N$ of `weak` is not total, where $\vec{d_i} = d_{i,1} \dots d_{i,m}$. 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 $d_{i,f(1)} \dots d_{i,f(n)}$ 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 $(d_{i,1} \dots d_{i,m}; \vec{a_i})$ to $(\vec{d_{i+1}}; \vec{a_{i+1}}) = (d_{i,f(1)}, \dots d_{i,f(n)}; \vec{a_i})$ 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 $\mathtt{S}$-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 $\mathtt{ap}$-rule, namely $\Pi(e_i) = \Gamma \vdash t[\vec{x}](u[\vec{x}]) : \vec{A} \to N$ is obtained from $\Gamma \vdash t[\vec{x}] : B \to \vec{A} \to N$ 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 $\mathtt{apvar}$.
8. The case of $\mathtt{cond}$-rule, namely $\Pi(e_i) = \mathtt{cond}(f[\vec{x}], g[\vec{x}, x]) : \Gamma \vdash C \to \vec{A} \to N$ 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, $\mathtt{cond}(f[\vec{d_i}], g[\vec{d_i}, x])m\vec{a'} : N$ 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 $\mathtt{cond}$ 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 $\pi = (e_1, e_2, \dots)$ 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 $\mathtt{CT}$-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