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

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

Abstract

A motivation for having a circular syntax for Gödel System $\mathbb{T}$ is that terms in a circular syntax are much shorter than the equivalent terms written in $\mathbb{T}$, yet they can be checked mechanically.

Several circular syntax for $\mathbb{T}$ (that is, infinite regular terms with the global trace condition) have been proposed, like CT by Anupam ([1], **S**4.5, page 20). In this paper we explore the possibility of defining a circular syntax directly for λ-abstraction and simple types, which we call CT-λ, instead of interpreting λ-abstraction through combinators first, then inserting a global trace condition afterwards. A circular syntax using binders instead of combinators could be more familiar for researchers working in the field of Type Theory. Again for reason of simplicity of use, we drop the restriction of having finitely many variables in infinite terms, and finitely many redexes, a restriction considered in previous works on the same topic, and we explore the consequences of ths choice.

We introduce our circular syntax as a fixed point operator defined by cases through a test on 0. We prove for CT-λ some of the results proved by Anupam for his combinatorial circular syntax: *(i)* every closed term of type N reduces to some numeral; *(ii)* strong normalization for all reductions sequences never reducing inside a fixed point operator; *(iii)* the equivalence between the set of terms of Gödel system $\mathbb{T}$ and the set of terms of CT-λ.

A consequence of the fact that we allow infinitely many variables for our infinite terms is that some other results proved by Anupam, namely Strong Normalization for all reductions and Church Rosser ([1], Theorem 51, page 35), fail. However, we recover for CT-λ the same properties in the limit: *(iv)* existence of a limit for all infinite reductions; *(v)* strong normalization in the limit for all fair reductions; *(vi)* Church-Rosser in the limit.

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 of terms and play a minor role in our paper.

The terms of $\mathbb{\Lambda}$ 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 term variables and no type variables, then the trees in $\mathbb{\Lambda}$ 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 (and $\mathbb{T}$, of course)

We introduce more sets of terms only used as intermediate steps in the definition of the semantic GTC and the syntax CT-λ.

1. WTyped $\subseteq \mathbb{\Lambda}$, the set of well-typed terms, is the of set terms having a unique type
2. Reg is the set of terms of $\mathbb{\Lambda}$ 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 possibly non-recursive, as we said). Terms of GTC denote total functionals.

We will prove that CT-λ is a decidable subset of Reg. CT-λ is a new variant of the existing 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. As we anticipated, 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 $\mathbb{\Lambda}$ of infinite circular terms, the subset `WTyped` of well-typed terms and a reduction relation for them. Infinite terms are labeled binary trees, therefore we have to define binary trees first, and before them the lists on $\{1,2\}$ and more related notions.

2.1 Lists and Binary Trees

Definition 2.1 (Lists on $\{1,2\}$) *1. We denote with* $\mathrm{List}(\{1,2\})$ *the set of all lists of elements of* $\{1,2\}$*:* $(), (1), (1), (1,1), (1,2), \ldots$*. We call the elements of* $\mathrm{List}(\{1,2\})$ *just lists for short.*

2. *If* $i_1, \ldots, i_n \in \{1,2\}$*, we write* $(i_1, \ldots, i_n)$ *for the list with elements* $i_1, \ldots, i_n$*.*
3. *We write* `nil` *for the empty list, or* $()$*.*
4. *If* $l = (i_1, \ldots, i_n)$*,* $m = (j_1, \ldots, j_m)$ *and* $l, m \in \mathrm{List}(\{1,2\})$ *we define*

$$l \star m = (i_1, \ldots, i_n,\ j_1, \ldots, j_m) \in \mathrm{List}(\{1,2\})$$

5. *The prefix order* $\leq$ *on lists is defined by* $l \leq m$ *if and only if* $m = l \star m'$ *for some* $m' \in \mathrm{List}(\{1,2\})$*.*

Binary trees are represented by set of lists.

Definition 2.2 (Binary trees) *1. A binary tree, just a tree for short, is any set* $T \subseteq \mathrm{List}(\{1,2\})$ *of lists including the empty list and closed by prefix:* `nil` $\in T$ *and for all* $l, m \in \mathrm{List}(\{1,2\})$ *if* $m \in T$ *and* $l \leq m$ *then* $l \in T$*.*

2. `nil` *is called the root of* T*, any* $l \in T$ *is called a node of* T *and if* $l \in T$*, then any* $l \star (i) \in T$ *is called a child of* l *in* T*.*
3. *A node* l *of* T *is a leaf of* T *if* l *is an maximal element of* T *w.r.t. the prefix order (i.e., if* l *has no children).*
4. *If* T *is a tree and* $l \in T$ *we define* $T\lceil l = \{m \in \mathrm{List}(\{1,2\}) | l \star m \in T\}$*.* $T\lceil l$ *is a tree and we say that* $T\lceil l$ *is a subtree of* T *in the node* l*, and for each* $m \in T\lceil l$ *we say that* $l \star m$ *is the corresponding node in* T*.*
5. *If* $l = (i)$ *we say that* T_l *is an immediate sub-tree of* T*.*
6. *A binary tree labeled on a set* L *is any pair* $\mathcal{T} = (T, \phi)$ *with* $\phi : T \to L$*. We call* $|\mathcal{T}| = T$ *the set of nodes of* $\mathcal{T}$*. For all* $l \in T$ *we call* $\mathtt{Lbl}(\mathcal{T}, l) = \phi(l)$ *the label of* l *in* $\mathcal{T}$*.*
7. *We define* $\mathcal{T}\lceil l = (T\lceil l, \phi_l)$ *and* $\phi_l(m) = l \star m$*. We call any* $\mathcal{T}\lceil l$ *a labeled subtree of* $\mathcal{T}$ *in the node* l*, an immediate sub-tree if* $l = (i)$*.*

The labeling of a node $m \in |(\mathcal{T}\lceil l)|$ is the same label of the corresponding node $l \star m$ of $|\mathcal{T}|$.
Let $n = 0, 1, 2$. Assume $\mathcal{T}_1, \ldots, \mathcal{T}_n$ are trees labeled on L and $l \in L$. Then we define

$$\mathcal{T} = l(\mathcal{T}_1, \ldots, \mathcal{T}_n)$$

as the unique tree labeled on L with the root labeled l and with: $\mathcal{T}\lceil (i) = \mathcal{T}_i$ for all $1 \leq i \leq n$.

2.2 Types, Terms and Contexts

Now we define the types of $\mathbb{\Lambda}$, then the terms and the contexts of $\mathbb{\Lambda}$.

Definition 2.3 (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, just types for short.*
2. `Type` *is the set of simples types.*
3. *We suppose be given a set* `var`*, consisting of all pairs* $x^T = (x, T)$ *of a variable name* x *and a type* $T \in$ `Type`*.*

Terms of $\mathbb{\Lambda}$ are labeled binary trees.

Definition 2.4 (Terms of $\mathbb{\Lambda}$) *The terms of $\mathbb{\Lambda}$ are all binary trees t with set of labels $L = \{x^T, \lambda x^T., \mathtt{ap}, 0, \mathtt{S}, \mathtt{cond}\}$ such that:*

1. *a node l labeled x^T or 0 is a leaf of t (no children).*
2. *a node l labeled $\lambda x^T.$ or $\mathtt{S}$ has a unique child $l\star(1)$.*
3. *a node labeled $\mathtt{ap}$, $\mathtt{cond}$ has two children $l\star(1)$, $l\star(2)$.*

We say that t is a sub-term of u if t is a labeled sub-tree of u, an immediate subterm if it is an immediate sub-tree.

Assume $l \in L = \{x^T, \lambda x^T., \mathtt{ap}, 0, \mathtt{S}, \mathtt{cond}\}$. We use the operation $l(\mathcal{T}_1 \dots \mathcal{T}_n)$ on labeled trees to define new terms in $\mathbb{\Lambda}$. If $t, u \in \mathbb{\Lambda}$ then $x^T, 0, \lambda x^T.t, \mathtt{ap}(t,u), \mathtt{S}(t), \mathtt{cond}(t,u) \in \mathbb{\Lambda}$ are terms with the root labeled l, and with immediate subterms among t, u. Conversely, each $v \in \mathbb{\Lambda}$ is in one of the forms: $x^T, 0, \lambda x^T.t$, $\mathtt{ap}(t,u)$, $\mathtt{S}(t)$, $\mathtt{cond}(t,u)$ for some $t, u \in \mathbb{\Lambda}$.

Now we define the regular terms $\mathbb{\Lambda}$.

Definition 2.5 (Regular terms of $\mathbb{\Lambda}$)

1. *We write $\mathtt{SubT}(t)$ for the (finite or infinite) set of subterms of t. Subterms are coded by the nodes of t, different nodes can code the same subterm.*
2. *$\mathtt{Reg}$ 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.*
3. *As usual, we abbreviate $\mathtt{ap}(t,u)$ with $t(u)$.*
4. *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 in $\mathbb{\Lambda}$.*
5. *A variable x^T is free in t if there is some $l \in t$ labeled x^T in T, and no $m < l$ labeled $\lambda x^T.$ in t. $\mathrm{FV}(t)$ is the set of free variables of t.*

$\mathtt{Num}$ is the representation inside $\mathbb{\Lambda}$ of the set $\mathbb{N}$ of natural numbers. All numerals are finite trees 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. Even if $\mathtt{SubT}(t)$ is finite, t can be an infinite tree.

We do not assume implicit renaming of bound variables, namely α-equivalence because the typing system (given later) does not have renaming rule as a primitive rule.

Example 2.1 *An example of regular term which is an infinite tree: the term $t = \mathtt{cond}(0,t) \in \mathbb{\Lambda}$. The set $\mathtt{SubT}(t) = \{t, 0\}$ of subterms of t is finite, therefore t is a regular term. However, t is an infinite tree (it includes itself as a subtree). The immediate sub-term chains of t are all $(t,t,t,\dots,t)$ and $(t,t,t,\dots,0)$. There is a unique infinite sub-term chain, which is $(t,t,t,\dots)$.*

In order to define the type of an infinite term, we first define contexts and sequences for any term of $\mathbb{\Lambda}$. A context is a list of type assignments to variables: our variables already have a type superscript, so in fact a type assignment $x^T : T$ 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.

Definition 2.6 (Contexts of $\mathbb{\Lambda}$)

1. *A context of $\mathbb{\Lambda}$ is any finite list $\Gamma = ({x_1}^{A_1} : A_1, \dots, x_n^{A_n} : A_n)$ of pairwise distinct variables, each assigned to its type superscript $A_1, \dots, A_n \in \mathtt{Type}$.*
2. *We denote the empty context with $\mathtt{nil}$. We write $\mathtt{Ctxt}$ 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 $\mathtt{Seq} = \mathtt{Ctxt} \times \mathtt{Type}$ for the set of all sequents.*
4. *A typing judgement is the list of a context Γ, a term t and a type A, which we write as $\Gamma \vdash t : A$. We write $\mathtt{Stat} = \mathtt{Ctxt} \times \mathbb{\Lambda} \times \mathtt{Type}$ for the set of all typing judgements.*
5. *We write $\mathrm{FV}(\Gamma) = \{{x_1}^{A_1}, \dots, {x_n}^{A_n}\}$. We say that Γ is a context for $t \in \mathbb{\Lambda}$ and we write $\Gamma \vdash t$ if $\mathrm{FV}(t) \subseteq \mathrm{FV}(\Gamma)$.*
6. *If $\Gamma = ({x_1}^{A_1} : A_1, \dots, x_n^{A_n} : A_n)$, $\Gamma' = (x'_1 : A'_1, \dots, x'_n : A'_{n'})$ are context of $\mathbb{\Lambda}$, then we write*
   (1) $\Gamma \subsetneq \Gamma'$ if for all $(x^A : A)$: $(x^A : A) \in \Gamma \Rightarrow (x^A : A) \in \Gamma'$

(2) $\Gamma \sim \Gamma'$ *if for all* $(x^A : A)$*:* $(x^A : A) \in \Gamma \Leftrightarrow (x^A : A) \in \Gamma'$

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 *for all* $x_i^{A_i} = x^T$*. 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. We do not identify two contexts which are one a permutation of the other, therefore we will need a rule to move from one to the other.

We define the term $t[u/x]$ obtained by substituting the infinite term u for the free variable x in t avoiding variable captures. In order to define the capture avoiding substitution under the assumption of not having implicit α-equivalence, we implicitly fix a well-order on variables and define the simultaneous substitution, written $t[u_1/x_1, \ldots, u_n/x_n]$ or $t[\vec{u}/\vec{x}]$, namely it satisfies the following:

- $x_i[u_1/x_1, \ldots, u_n/x_n] = u_i$ and $y[\vec{u}/\vec{x}] = y$, where $y \notin \vec{x}$.
- $(f(a))[\vec{u}/\vec{x}] = f[\vec{u}/\vec{x}](a[\vec{u}/\vec{x}])$.
- $(\lambda z^T.b)[\vec{u}/\vec{x}] = \begin{cases} \lambda z^T.(b[(\vec{u}/\vec{x})_{\overline{z}}]), & \text{if } z \notin \mathrm{FV}(\vec{u}), \\ \lambda z'^T.(b[(\vec{u}/\vec{x})_{\overline{z}}, z'/z]), & \text{otherwise,} \end{cases}$
  where $(\vec{u}/\vec{x})_{\overline{z}}$ is the one obtained by removing u/z (for some u) from $\vec{u}/\vec{x}$, and z' is the least variable (with respect to the well-order) such that $z' \notin \mathrm{FV}(b, \vec{u})$.
- $0[\vec{u}/\vec{x}] = 0$.
- $\mathtt{S}(t)[\vec{u}/\vec{x}] = \mathtt{S}(t[\vec{u}/\vec{x}])$.
- $\mathtt{cond}(f, g)[\vec{u}/\vec{x}] = \mathtt{cond}(f[\vec{u}/\vec{x}], g[\vec{u}/\vec{x}])$.

Formally we define the substitution as follows.

Definition 2.7 (Substitution for terms of $\mathbb{A}$) *A set* $\theta = \{u_1/x_1, \ldots, u_n/x_n\}$ *is called a substitution if* $x_1, \ldots, x_n$ *are pairwise distinct variables. It is called renaming if* $u_1, \ldots, u_n$ *are pairwise distinct variables. We define* $\theta(y)$ *by* u_i *if* $y = x_i$*, and by* y *otherwise. We write* θ, θ' *if* $\theta \cup \theta'$ *is a substitution. The substitution* $\theta_{\overline{x}}$ *is defined as the one obtained by removing* u/x *(for some* u*) from* θ*.*

Let t *be a term of* $\mathbb{A}$*, which is the labeled tree* (T_t, ϕ_t)*, and* θ *be* $\{u_1/x_1, \ldots, u_n/x_n\}$*, where each* u_i *is* (T_{u_i}, ϕ_{u_i})*. Then the term* $t[\theta]$ *is* (T, ϕ)*, where* $T = T_t \cup \{l \star m \mid \phi_t(l) = x_i \in \mathrm{FV}(t) \text{ and } m \in T_{u_i}\}$*. For each* $l \star m$ *in the latter part,* $\phi(l \star m) = \phi_{u_i}(m)$*. For* $l \in T_t$*, we inductively define* $\phi(l)$ *and* θ_l *(such that* $\theta_l = \theta', \theta_{ren}$*, where* $\theta' \subseteq \theta$ *and* θ_{ren} *is a renaming) as follows.*

- $\theta_{\mathtt{nil}} = \theta$*.*
- *If* $\phi_t(l) = y^T \notin \mathrm{FV}(t)$*, then* $\phi(l) = \theta_l(y^T)$*.*
- *If* $\phi_t(l) = 0$*, then* $\phi(l) = 0$*.*
- *If* $\phi_t(l) = \mathtt{S}$*, then* $\phi(l) = \mathtt{S}$ *and* $\theta_{l\star(1)} = \theta_l$*.*
- *If* $\phi_t(l) \in \{\mathtt{ap}, \mathtt{cond}\}$*, then* $\phi(l) = \phi_t(l)$ *and* $\theta_{l\star(1)} = \theta_{l\star(2)} = \theta_l$*.*
- *If* $\phi_t(l) = \lambda z^T.$ *and* $z \notin \mathrm{FV}(\vec{u})$*, then* $\phi(l) = \lambda z^T.$ *and* $\theta_{l\star(1)} = (\theta_l)_{\overline{z}}$*.*
- *If* $\phi_t(l) = \lambda z^T.$ *and* $z \in \mathrm{FV}(\vec{u})$*, then* $\phi(l) = \lambda z'^T.$ *and* $\theta_{l\star(1)} = (\theta_l)_{\overline{z}}, z'/z$*, where* z' *is the least variable such that* $z' \notin \mathrm{FV}(t\lceil l\star(1), \vec{u})$*.*

We often abbreviate $t[\{u_1/x_1, \ldots, u_n/x_n\}]$ *by* $t[u_1/x_1, \ldots, u_n/x_n]$*. For a context* $\Gamma = (x_1 : A_1, \ldots, x_n : A_n)$ *and a renaming* θ*, we write* $\Gamma[\theta]$ *for* $(\theta(x_1) : A_1, \ldots, \theta(x_n) : A_n)$ *if it is a context.*

We have the following substitution lemma.

Lemma 2.8 (Substitution lemma) $t[u/x][v/y] = t[t/y][u[v/y]/x]$ *if* $x \notin \mathrm{FV}(v)$*.*

Proof (FILL THIS)

2.3 Typing Rules

We define typing rules for terms of $\mathbb{\Lambda}$ and the subset $\mathtt{WTyped}$ 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 the *typing rule* $\mathtt{ap}$ *for an application* $t(u)$, which we split in two sub-rules, $\mathtt{ap}_v$ and $\mathtt{ap}_{\neg v}$, according if u is a variable or is not a variable. We need to insert this extra information t in typing because it is important for checking termination of the computation of $t(u)$, as we will explain later.

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, but we we split the typing rule for $\mathtt{ap}$ into two sub-cases, $\mathtt{ap}_v$ and $\mathtt{ap}_{\neg v}$.

We have a a single structural rule $\mathtt{weak}$ for extending a context Γ to a context $\Gamma' \supsetneq \Gamma$. When $\Gamma' \sim \Gamma$ (when Γ', Γ are permutation each other), the rule $\mathtt{weak}$ 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 renaming. The lack of a renaming rule is a *difference with the circular syntax for inductive proofs*.

The reason of not having implicit α-conversion mentioned in the previous section comes from this assumption. For example, if we identify the α-equivalent terms $\lambda x^A.x$ and $\lambda z^A.z$, the left derivation is a correct proof, but the right one is not ($(z:A, z:A)$ is not a context). That is, the proof structure does not closed under identity of the α-equivalence.

$$\frac{\dfrac{}{z:A, x:A \vdash x:A}}{z:A \vdash \lambda x^A.x : A \to A} \qquad \frac{\dfrac{}{z:A, z:A \vdash z:A}}{z:A \vdash \lambda z^A.z : A \to A}$$

The provability of sequents is closed under the variable renaming and the α-equivalence. We will discuss this point later.

Definition 2.9 (Typing rules of $\mathbb{\Lambda}$) *Assume* $\Gamma = {x_1}^{A_1} : A_1, \ldots, {x_n}^{A_n} : A_1$ *is a context. Let* $p = 0, 1, 2$.

1. *A rule is a list of* $p+1$ *typing judgements:* $\Gamma \vdash t_1 : A_1, \ldots, \Gamma \vdash t_p : A_p, \Gamma \vdash t : A$. *The first* p *sequents are the premises of the rule (at most two in our case), the last sequent is the conclusion of the rule.*
2. *We read the rule above: "if* $\Gamma \vdash t_1 : A_1, \ldots, \Gamma \vdash t_p : A_p$ *then* $\Gamma \vdash t : A$*".*

We list the typing rules of $\mathbb{\Lambda}$*: the rules* $\mathtt{ap}_v$ *and* $\mathtt{ap}_{\neg v}$ *are restricted.*

1. $\mathtt{weak}$*-rule (Weakening + Exchange). If* $\Gamma \vdash t : T$ *and* $\Gamma \subsetneq \Gamma'$ *then* $\Gamma' \vdash t : T$
2. $\mathtt{var}$*-rule. If* $x^A \in \Gamma$ *then* $\Gamma \vdash x^A : A$.
3. λ*-rule. If* $\Gamma, x^A : A \vdash b : B$ *then* $\Gamma \vdash \lambda x^A.b : A \to B$.
4. $\mathtt{ap}_v$*-rule. If* $\Gamma \vdash f : A \to B$ *then* $\Gamma \vdash f(x^A) : B$*, provided* $(x^A : A) \in \Gamma$.
5. $\mathtt{ap}_{\neg v}$*-rule. If* $\Gamma \vdash f : A \to B$ *and* $\Gamma \vdash a : A$ *then* $\Gamma \vdash f(a) : B$*, provided* a *is not a variable*
6. 0*-rule.* $\Gamma \vdash 0 : N$
7. $\mathtt{S}$*-rule. If* $\Gamma \vdash t : N$ *then* $\Gamma \vdash \mathtt{S}(t) : N$.
8. $\mathtt{cond}$*-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* $\mathbb{\Lambda}$ *as*

$$\mathtt{Rule} = \{r \in \mathtt{Seq} \cup \mathtt{Seq}^2 \cup \mathtt{Seq}^3 | r \text{ instance of some rule of } \mathbb{\Lambda}\}$$

A rule is uniquely determined from its conclusion, provided we know whether the rule is a weakening or not. Two rules are equal if they have the same assumptions and the same conclusion.

Proposition 2.10 (Rules and subterms) *Assume* $r, r' \in \mathtt{Rule}$ *have conclusion* $\Gamma \vdash t : A$, $\Gamma' \vdash t' : A'$ *respectively.*

1. *If* r, r' *are non-weakening rules and* $\Gamma \vdash t : A = \Gamma' \vdash' t' : A'$ *then* $r = r'$.
2. *If* r *is not a weakening and the premises of* r *are some* $\Gamma_1 \vdash t_1 : A_1, \ldots, \Gamma_p \vdash t_p : A_p$*, then the list of immediate subterms of* t *is exactly* $t_1, \ldots, t_p$.

From the typing rules we define the proofs that a term is well-typed. Proofs are binary trees on the set `Rule`, each node is associated to a typing judgement which is a consequence of the typing judgements associated to the children node under some rule $r \in$ `Rule` of $\mathbb{\Lambda}$.

The next definition is the formal definition of proof. We mention that we will introduce a restriction we call *Almost-left-finite* on proof trees. The restriction Almost-left-finite is used to prevent trivial proofs. For instance, we want to forbid a proof cyclically switching the variables x and y in the context of a term, by an infinite nesting of `weak`-rule (Weakening + Exchange rule):

$$\dfrac{\dfrac{\vdots}{(y^U : U, x^T : T) \vdash t : A}\ \texttt{weak}}{(x^T : T, y^U : U) \vdash t : A}\ \texttt{weak}$$

Another example: we want to forbid an infinite left nesting of applications, as in a proof of $t_n : A^n \to A$ where $t_n = t_{n+1}(t_0)$:

$$\dfrac{\dfrac{\cdots}{\vdash t_{n+1} : A^{n+1} \to A} \quad \dfrac{\cdots}{\vdash t_0 : A}}{\vdash t_n : A^n \to A}\ \texttt{ap}_{\neg v}$$

These proof trees are meaningless and should be ruled out. The Almost-Left-Finite condition will ask that all leftmost branches in any sub-proof are finite, say, that no infinite leftmost branch made of `weak`-rules or of $\texttt{ap}_{\neg v}$ exists. The idea is that when the main operation in a programming language is application, then the leftmost branch uniquely determines the type of the term and therefore it should be finite.

Definition 2.11 (Well-typed term of $\mathbb{\Lambda}$) *Assume $\Pi = (T, \phi)$ is a binary tree labeled on* `Rule`*. Assume Γ is a context of t (i.e, $\mathrm{FV}(t) \subseteq \Gamma$) and $A \in$* `Type`

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

1. *$\phi(\texttt{nil}) = \Gamma \vdash t : A$.*
2. *Assume that*

   (1) $l \in T$ is labeled with $\phi(l) = \Delta \vdash u : B$.

   (2) The children of l in T are labeled $\phi(l\star(1)) = \Delta_1 \vdash u_1 : B_1, \ldots, \phi(l\star(p)) = \Delta_p \vdash u_p : B_p$

   Then for some rule $r \in$ `Rule` of $\mathbb{\Lambda}$ we have

$$\dfrac{\Delta_1 \vdash u_1 : B_1 \quad \cdots \quad \Delta_p \vdash u_p : B_p}{\Delta \vdash u : B}\ r$$

3. *A path π in a proof tree Π is almost-leftmost if there are only finitely many rules with two premises in π such that π does not choose the leftmost premise.*
4. *A proof Π is Almost-left-finite if all almost-lefmost paths π in Π are finite*

We can check that a proof is almost-left-finite if and only if all leftmost paths of all sub-proofs are finite.

In the case of a regular proof, we can decide in polynomial time in the number of sub-proofs whether a proof is almost-left-finite. Here is the algorithm. Suppose a possibly infinite proof has n subproofs. For each sub-proof we follow the leftmost path, after n steps either we reach some rule without premises (either a `var`-rule or a 0-rule), or we loop. Then:

1. If for all sub-proofs we stop then the proof is almost-left-finite
2. If for some sub-proof we loop, then there is infinite leftmost path and the proof is *not* almost-left-finite.

We estimate the computation time to $O(n)$ for each of the n sub-proofs, and the total computation time to $O(n^2)$.

Eventually we define the well-typed terms of $\mathbb{\Lambda}$ using almost-left-finite proofs.

Definition 2.12 (Well-typed term of $\mathbb{\Lambda}$)

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 and only if* $\Pi : \Gamma \vdash t : A$ *for some almost-left-finite proof* Π*, for some* A *and some context* $\Gamma \supseteq \mathrm{FV}(t)$.
3. `WTyped` *is the set of well-typed* $t \in \mathbb{\Lambda}$.

Well-typed terms are closed by substitution.

We prove that the type assigned to a term by an Almost-Left-Finite proof is *unique*: if two Almost-Left-Finite proofs assign two types T, T' to the same possibly infinite term t, then $T = T'$.

Proposition 2.13 (Uniqueness of almost-left-finite type) *If* $t \in \mathbb{\Lambda}$, $\Pi : \Gamma \vdash t : A$ *and* $\Pi' : \Gamma' \vdash t : A'$ *are two almost-left-finite proofs then* $A = A'$.

Proof By induction on the sum of the length of the leftmost branch in Π and Π'. These lengths are finite because Π and Π' are assumed to be almost-left-finite. Let r be the last rule of Π and r' be the last rule of Π'.

1. Assume $r = \mathtt{weak}$. Then Π is obtained from a proof $\Pi_1 : \Gamma_1 \vdash t : A$. By induction hypothesis on Π_1 and Π' we conclude that $A = A'$.
2. Assume $r \neq \mathtt{weak}$ and $r' = \mathtt{weak}$. Then Π' is obtained from a proof $\Pi'_1 : \Gamma_1 \vdash t : A$. By induction hypothesis on Π and Π'_1 we conclude that $A = A'$.
3. Assume $r \neq \mathtt{weak}$ and $r' \neq \mathtt{weak}$. Then both r and r' are the typing rule for the outermost constructor of t and for some $h = 0, 1, 2$ the proof Π has premises $\Pi_1, \ldots, \Pi_h$ and the proof Π' has premises $\Pi'_1, \ldots, \Pi'_h$ (for the same h). We distinguish one case for each constructor.
   (1) Assume $t = x^T$. Then $r = r' = \mathtt{var}$-rule and $A = A' = T$.
   (2) Assume $t = 0^N$. Then $r = r' = 0$-rule and $A = A' = N$.
   (3) Assume $t = \mathtt{S}(u)$. Then $r = r' = \mathtt{S}$-rule and $\Pi_1 : \Gamma \vdash u : N$ and $\Pi_2 : \Gamma' \vdash u : N$ and $A = A' = N$.
   (4) Assume $t = f(u)$. Then $r = r' = \mathtt{ap}$-rule and $\Pi_1 : \Gamma \vdash f : U \to A$ and $\Pi_2 : \Gamma' \vdash f : U' \to A'$. By induction hypothesis on Π_1 and Π'_1 we deduce that $(U \to A) = (U' \to A')$, in particular that $A = A'$.
   (5) Assume $t = \lambda x^T.b$. Then $r = r' = \lambda$-rule and $\Pi_1 : \Gamma \vdash b : B$ and $\Pi_2 : \Gamma' \vdash b : B'$. By induction hypothesis on Π_1 and Π'_1 we deduce that $B = B'$, in particular that $A = T \to B = T \to B' = A'$.
   (6) Assume $t = \mathtt{cond}(f, g)$. Then $r = r' = \mathtt{cond}$-rule and $\Pi_1 : \Gamma \vdash f : A$ and $\Pi_2 : \Gamma' \vdash f : A'$. By induction hypothesis on Π_1 and Π'_1 we deduce that $A = A'$.

We provide some examples of well-typed and not well-typed terms. Some term in $\mathbb{\Lambda}$ has no type, like the application $0(0)$ of the non-function 0.

Example 2.2 *We provide a term in* $\mathbb{\Lambda}$ *having more than one type. Let* $t = u(0)$ *and* $u = \mathtt{cond}(t, u)$. t *has an infinite lefmost branch* $t, u, t, u, \ldots$*, therefore* t *is not almost-left-finite and it has no almost-left-finite typing proof. Unicity fails and we can prove* $\vdash t : A$ *for all types* $A \in \mathtt{Type}$. *The subterms of* t *are* $\{t, u, 0\}$ *and a proof* $\Pi : \emptyset \vdash t : A$ *is*

$$\dfrac{\dfrac{\begin{array}{c}\vdots\\ \vdash t : A\end{array} \qquad \dfrac{\begin{array}{c}\vdots\\ \vdash u : N \to A\end{array}}{\vdash u : N \to A}\ \mathtt{cond}}{\vdash t : A}\ \mathtt{ap}_{\neg v} \qquad \dfrac{}{\vdash 0 : N}\ 0}{\vdash t : A}$$

Formally, the typing proof $\Pi = (T, \phi) :\vdash t : A$ *is defined by* $|\Pi| = |t|$ *and:*

1. $\mathtt{Lbl}(\Pi, l) = (\vdash A)$ *for all* $l \in |t|$ *such that* $t\lceil l = t$,
2. $\mathtt{Lbl}(\Pi, l) = (\vdash N \to A)$ *for all* $l \in |t|$ *such that* $t\lceil l = u$,
3. $\mathtt{Lbl}(\Pi, l) = (\vdash N)$ *for all* $l \in |t|$ *such that* $t\lceil l = 0$.

A term with at least two types has the leftmost branch infinite, as it is the case for t above. We proved that if *all* subterms of a term have the leftmost branch finite, then the term has at most one type.

Claim (existence of a polynomial-time typing algorithm). We claim without proof that if a term t is regular then we can decide if an almost-left-finite proof $\Pi : \Gamma \vdash A$ exists. If an almost-left-finite proof Π exists then at least one of almost-left-finite proofs is regular and we can compute it. We first check whether the term is almost-left-finite in time $O(n^2)$. If it is not we reject the term. Otherwise we start the standard recursive typing algorithm, until a type has been inferred consistently for each of the finitely many subterms, or some type inconsistency has been found. In the first case we accept the term and we return the time, in the second case we reject the term and we return no type. The computation requires quadratic time in the size of the graph representing the term.

2.4 Reduction Rules

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

1. $\mapsto_\beta$*:* $(\lambda x^A.b)(a) \mapsto_\beta b[a/x]$
2. $\mapsto_{\mathtt{cond}}$*:* $\mathtt{cond}(f,g)(0) \mapsto_{\mathtt{cond}} f$ *and* $\mathtt{cond}(f,g)(\mathtt{S}(t)) \mapsto_{\mathtt{cond}} g(t)$.
3. $\mapsto$ *is the contraction of one* β *or* `cond` *redex: the context closure of* $\mapsto_\beta$ *and* $\mapsto_{\mathtt{cond}}$.
4. $\mapsto^*$ *is the reflexive and transitive closure of* $\mapsto$ *(zero or more reductions).*
5. $\mapsto^+$ *is the transitive closure of* $\mapsto$ *(one or more reductions).*
6. *The* `cond`*-depth of a node* $l = (i_1, \ldots, i_n) \in t$ *is the number of* $1 \leq h < n$ *such that* $(i_1, \ldots, i_h)$ *has label* `cond` *and* $i_{h+1} = 2$ *(the number of* `cond` *such that* l *occurs in the right-hand-side of such* `cond`*).*
7. *The* n*-safe level of* t *is the set of nodes of* t *of depth* $\leq n$.
8. *We say that* $t \mapsto_{n,\mathtt{safe}} u$*, or that* t n*-safely reduces to* u*, if we reduce* t *to* u *by contracting a single redex in the* n*-safe level of* t *(i.e. of* `cond`*-depth* $\leq n$*).*
9. *A term is* n*-safe-normal if all its redexes (if any) have* `cond`*-depth* $> n$.
   *We abbreviate "*0*-safe normal" with just "safe normal".*

Example 2.3 *Let* $u = \mathtt{cond}.(0, (\lambda z.u)(z))$*, where we omitted the type superscript of* z *because it is irrelevant. Then* u *is* 0*-safe-normal (safe normal for short), because all redexes in* u *are of the form* $(\lambda z.u)(z)$ *and are in the right-hand-side of a* `cond`*. However* u *has infinitely many nodes which are* β*-redexes. Indeed, the tree form of* u *has the following branch:*

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

This branch is cyclic, infinite, and it includes infinitely many β*-redexes, all equal to the same term* $(\lambda z.u)(z)$.

Through branches crossing the right-hand-side of some `cond` we will-express fixed-point equations. Reductions on fixed-point equations can easily loop. However, in order to produre the output of a computation we only need to reduce closed terms of type N to a numeral. We will prove that for this taks we only need 0-safe reductions, those in the the right-hand-side of *no* `cond`, and that 0-safe reductions strongly normalize.

Example 2.4 *This is an example of safe* `cond`*-reductions from a term* $v(n)$ *to a normal form. Assume* $n \in \mathtt{Num}$ *is any numeral and* $v = \mathtt{cond}(1, v)$*. There are only finitely many reductions from* $v(n)$*, and they are all* 0*-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* 1 *and we stop. Thus, the term* $v(n)$ *strongly normalizes to* 1 *in* $(n+1)$*-steps, in fact* $v(n)$ *has a unique reduction path, having length* $(n+1)$ *and terminating in* 1.

There are terms without a normal form, but we can get as close as we want to a normal form In fact, we will prove that for all n, all terms have some n-safe normal form, that is, with no redexes of `cond`-depth $\leq n$. The n-safe normal form of a term is unique up n-safe equality: any two n-safe normal forms have the same n-safe level, though they may be different in some level of `cond`-depth $> n$.

We will also prove that terms have a limit normal form, obtained by taking the limit of the n-safe normal forms of the term, and that this limit normal form is unique.

3 The trace of the infinite λ-terms

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 notion of *connection* between arguments of type N in the proof that t is well-typed. To this aim, we need the notions 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, while B_3 is an unnamed argument (it will be denoted by some bound variable in t). The index of an N-argument of t is any $j \in \{1, 2, 3\}$ such that $A_j = N$ or $B_j = N$ respectively: in this case all integers $1, 2, 3$ are indexes of some N-argument, in general we can have argument with type different from N.

Remark that for an open term $t[x_1, x_2]$ we list among the "argument types" also the types A_1, A_2 of the free variables. We motivate this 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 A_1, A_2, B_3 and they include A_1, A_2.

Below is the formal definition of argument types for a term.

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* N*-argument of* t *is any* $j \in \{1, \ldots, n+m\}$ *such that* $C_j = N$.

We now define the connection between N-arguments of subterms of t in a proof $\Pi : t : \Gamma \vdash A$ of $\mathbb{\Lambda}$. The connection describes how an input moves through the infinite unfolding of the term. 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 N-argument which are in connection.

Example 3.1 An example of atom connection for some instance of the `weak`-rule.

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

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

Example 3.2 *An example of atom connection for some instance of the* $\mathtt{ap}_{\neg v}$*-rule. We assume that* a *is not a variable.*

$$\frac{x_1 : \mathbf{N}, x_2 : \mathbf{N} \vdash f : \mathbf{N}, \mathbf{N} \to N \quad x_1 : \mathbf{N}, x_2 : \mathbf{N} \vdash a : N}{x_1 : \mathbf{N}, x_2 : \mathbf{N} \vdash f(a) : \mathbf{N} \to N}\ (\mathtt{ap}_{\neg v})$$

Remark that the first unnamed argument of f (colored in **old gold**) is in connection with no argument in the rule $\mathtt{ap}_{\neg v}$. 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.

We postpone examples with the rule $\mathtt{ap}_v$.

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

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

Remark that the first unnamed argument of $\mathtt{cond}(f, g)$ (colored in **old gold**) is in connection the type of the first unnamed one of g in the second premise, but it has no connection with the first premise.

3.1 A formal definition of connection

The definition of connection requires to define an injection

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

We set $\mathtt{ins}(p, x) = x + 1$ if $x \geq p + 1$, and $\mathtt{ins}(p, x) = x$ if $x \leq p$. The role of $\mathtt{ins}$ is inserting one fresh index $p + 1$ for a new argument type higher in the proof connected to no argument in the conclusion.

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

For each N-argument k in t, for $t' = t$ or t' immediate subterm of t each N-argument k' in t' we define the relation: "k', t' the successor of k, t". We require:

1. *if t is obtained by a rule $\mathtt{weak}$ from $t' = t$, described by a map $\phi : \Gamma \to \Delta$, then $k = \phi(k')$.*
2. *if $t = f(a)$ is obtained by a rule $\mathtt{ap}_{\neg v}$ (i.e., a is not a variable) and $t' = f$ then $k' = \mathtt{ins}(n, k)$. If $t' = a$ then $k' = k$ and $k' \leq n$.*
3. *if $t = f(x_i^{A_i})$ is obtained by a rule $\mathtt{ap}_v$ and $t' = f$ then $k' = \mathtt{ins}(n, k)$ or $k' = n + 1$ and $k = i$.*
4. *if $t = \mathtt{cond}(f, g)$ and $t' = f$ then $k' = \mathtt{ins}(n, k)$. If $t' = g$ then $k' = k$.*

We require $k = k'$ in all other cases, which are: $\mathtt{S}$, λ.
No connection is defined for the rules $\mathtt{var}$, 0.

Below we include some examples. We draw in the same color two N-argument which are in connection. In the case of the rule $\mathtt{ap}_v$, an argument can be connected to two arguments above it.

Example 3.4 *Atom connection for $\mathtt{ap}_v$. We assume that x is a variable.*

$$\frac{x_1 : \mathbf{N}, x_2 : \mathbf{N}, x_3 : \mathbf{N} \vdash f : \mathbf{N}, \mathbf{N} \to N}{x_1 : \mathbf{N}, x_2 : \mathbf{N}, x_3 : \mathbf{N} \vdash f(x_3) : \mathbf{N} \to N}\ (\mathtt{ap}_v)$$

Remark that the last variable in the conclusion of the rule is in connection with the first unnamed argument of f (colored in **old gold**) in the premise, and with the variable with the same name in the premise: there are two connections.

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

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

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, when we move up in a $\mathtt{ap}_v$-rule, the type of last free variable corresponds to the type of the first unnamend argument. When we move up in a λ-rule it is the other way round.

The connection on N-arguments in a proof $\Pi : \Gamma \vdash t : A$ defines a graph $\mathtt{Graph}(\Pi)$ whose nodes are all pairs (k, u), with u subterm of t and k index of some N-argument of u. $\mathtt{Graph}(\Pi)$ is finite when t is regular. A trace represents the movement of an input information through the infinite unfolding of a tree. Formally, 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 $\mathbb{\Lambda}$) *Assume Π is any typing proof of $t \in \mathtt{WTyped}$.*

1. *A path of Π is any finite or infinite branch $\pi = (i_0, \dots, i_n, \dots)$ of Π.*
2. *Assume $\pi = (t_0, \dots, t_n, \dots)$ is a path of Π, finite or infinite. A finite or infinite trace τ of π in Π is a list $\tau = ((k_m, t_m), \dots, (k_n, t_n), \dots)$ such that for all $i = m, \dots, n, \dots$:*
   (1) k_i is an index of some N-argument of t_i
   (2) if $i + 1$ is an index of τ then (k_{i+1}, t_{i+1}) is connected with (k_i, t_i) in Π.

The starting point of a trace could be different from the starting point of the path to which the trace belongs.

4 The circular system CT-λ

We introduce a condition call *global trace condition* for all proofs Π that the term t is well-typed, then the subset GTC of the set WTyped of well-typed terms of $\mathbb{\Lambda}$ satisfying the global trace condition. GTC is a set of total recursive functionals. We use GTC as a semantics for the set CT-λ of circular λ-terms we will define.

CT-λ will be the regular terms of CT-λ. CT-λ is a decidable set. For the terms of CT-λ we will prove strong normalization, church-rosser for the "safe" part of a term, and the fact that every closed normal term of type N is a numeral. As a consequence, all terms CT-λ will be interpreted as total functionals.

From the N-argument connection we now define the global trace condition and the set of terms of GTC and of CT-λ.

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

1. τ *is progressing in* i *if* $t_i = \mathtt{cond}(f, g)$ *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 some typing proof* Π, *of* t, *for all infinite paths* π *of* t *in* Π, *there is some infinitely progressing path* τ *in* π *and* Π. GTC *is the set of well-typed terms* $t \in \mathbb{\Lambda}$ *satisfying the global trace condition.*
3. CT-λ = GTC$\cap$Reg*: the cyclic* λ*-terms of* CT-λ *are all well-typed terms which are regular trees (having finitely many subtrees), and which 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 and they form a non-countable sets. Therefore we would expect that the global trace condition is not decidable. Surprisingly, global trace 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 there is a typing proof which is a finite graph and for which the global trace condition is decidable.

We believe that the global trace condition is decidable quite fast for realistic λ-terms. Another nice feature of the global trace condition is that for $t \in$ GTC we require that there is some proof $\Pi : \Gamma \vdash t : A$ whose infinite paths all have some infinite progressing trace. This proof is almost-left-finite, because all leftmost paths from a sub-proof have no progress point and therefore are finite. We conclude that $t \in$ GTC implies that $t \in$ WTyped: we can assign a unique type to t with a "meaningful proof", that is, an almost-left-finite finite proof.

We include now some examples of cyclic λ-terms. We recall that they have to satisfy GTC, therefore they are almost-left-finite, and by definition they are regular.

4.1 The sum map

A first example of term of CT-λ. We provide an infinite regular term Sum computing the sum on N. In this example, the type superscript N of variables x^N, z^N is omitted.

Example 4.1 *We set* $\mathtt{Sum} = \lambda x.\mathtt{cond}(x, \lambda z.\mathtt{S}(\mathtt{Sum}(x)(z)))$. Sum *is a regular term because it has finitely many subterms:*

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

Sum *is well-typed by the following derivation, in which we have a back edge from the* † *above to the* †

below.

$$\dfrac{\dfrac{}{x : N \vdash x : N}\ \mathtt{var} \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\vdots}{\vdash \mathtt{Sum} : N, \mathbf{N} \to N \quad (\dagger)}}{x : N, z : N \vdash \mathtt{Sum} : N, \mathbf{N} \to N}\ \mathtt{weak}}{x : N, z : N \vdash \mathtt{Sum}(x) : \mathbf{N} \to N}\ \mathtt{ap}_v}{x : N, z : \mathbf{N} \vdash \mathtt{Sum}(x)(z) : N}\ \mathtt{ap}_v}{x : N, z : \mathbf{N} \vdash \mathtt{S}(\mathtt{Sum}(x)(z)) : N}\ \mathtt{S}}{x : N \vdash \lambda z.\mathtt{S}(\mathtt{Sum}(x)(z)) : \mathbf{N} \to N}\ \lambda}{\dfrac{x : N \vdash \mathtt{cond}(x, \lambda z.\mathtt{S}(\mathtt{Sum}(x)(z))) : \mathbf{N} \to N \quad (\spadesuit)}{\vdash \mathtt{Sum} : N, \mathbf{N} \to N \quad (\dagger)}\ \lambda}\ \mathtt{cond}$$

We colored in **old gold** one sequence of atoms $\mathbf{N}$: this is one trace of the proof. The colored trace is the unique infinite trace, it is cyclic and it is infinitely progressing. We marked ♠ the only progress point of the only infinite trace. The progress point is repeated infinitely many times.

4.2 The Iterator

Example 4.2 *We define a term* $\mathtt{It}$ *of* $\mathtt{CT}$-λ *computing the iteration of maps on* N*. The term* $\mathtt{It}$ *is a normal term of type* $(N \to N), N, N \to N$ *such that* $\mathtt{It}(f, a, n) = f^n(a)$ *for all numeral* $n \in \mathtt{Num}$*. We have to solve the equations:*

1. $\mathtt{It}(f, a, 0) \sim a$
2. $\mathtt{It}(f, a, \mathtt{S}(t)) \sim f(\mathtt{It}(f, a, t))$

where f*, a abbreviate* $f^{N \to N}$ *and* a^N*. We solve them with* $\mathtt{It} = \lambda f, a.\mathtt{it}$ *where*

$$\mathtt{it} = \mathtt{cond}(a, \lambda x.f(\mathtt{it}(x))) : N \to N$$

is a term in the context $\Gamma = (f : N \to N, a : N)$*.*

Proposition 4.2 $\mathtt{It} \in \mathtt{Reg} \cap \mathtt{WTyped} \cap \mathtt{GTC} = \mathtt{CT}$-$\lambda$.

Proof The term $\mathtt{It}$ is well-typed and regular by definition. We check the global trace condition.
We follow the unique infinite trace τ of the last unnamed argument N of $\mathtt{It}$ in the unique infinite path π of $\mathtt{It}$. The trace τ moves from $\mathtt{It} = \lambda f.\lambda a.\mathtt{it}$ to the first unnamed argument of the sub-term $\lambda a.\mathtt{it}$, then to $a : N$ in the context of $\mathtt{it} = \mathtt{cond}(a, \lambda x.f(\mathtt{it}(x)))$. Then the infinite path π and moves in this order to: $\lambda x.f(\mathtt{it}(x))$, $f(\mathtt{it}(x))$, $\mathtt{it}(x)$, $\mathtt{it}$ In the meanwhile, τ progresses from $\mathtt{cond}$ to the second argument $\lambda x.f(\mathtt{it}(x))$ of $\mathtt{cond}$, then moves to x in the context of $f(\mathtt{it}(x))$, then to x in the context $\mathtt{it}(x)$, and eventually to x in the context of $\mathtt{it}$. In this moment the infinite path π loops from $\mathtt{it}$ to $\mathtt{it}$. At each loop the trace τ progresses once. We conclude that τ infinitely progresses.

The proof above includes a type inference from the term $\mathtt{it}$ to itself. We draw the type inference in the picture below. We have a back edge from the † on the top to the † in the bottom, and we marked ♠ the only progress point, which is cyclically repeated in τ. We abbreviate $\Gamma = (f : N \to N, a : N)$.

$$\dfrac{\dfrac{}{\Gamma \vdash a : N}\ \mathtt{var} \qquad \dfrac{\dfrac{\dfrac{}{\Gamma, x : N \vdash f : N \to N}\ \mathtt{var} \qquad \dfrac{\dfrac{\vdots}{\Gamma \vdash \mathtt{it} : \mathbf{N} \to N \quad (\dagger)}}{\Gamma, x : N \vdash \mathtt{it} : \mathbf{N} \to N}\ \mathtt{weak}}{\Gamma, x : \mathbf{N} \vdash \mathtt{it}(x) : N}\ \mathtt{ap}_v}{\Gamma, x : \mathbf{N} \vdash f(\mathtt{it}(x)) : N}\ \mathtt{ap}}{\Gamma \vdash \lambda x.f(\mathtt{it}(x)) : \mathbf{N} \to N}\ \lambda}{\Gamma \vdash \mathtt{it} : \mathbf{N} \to N \quad (\spadesuit, \dagger)}\ \mathtt{cond}$$

4.3 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 `nil` and `cons`. Elimination rules are not required in our example.

Example 4.3 *We will define a term* `In` *with one argument* $f : N \to N$ *and three argument* $a, x, y : N$ *(we skip all type superscripts), such that*

$$\mathtt{In}(f, a, n, m) \ \sim \ [f^n(a), f^{n+1}(a), \dots, f^{n+m}(a)] \ : \ \alpha$$

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

$$\mathtt{In}(f, a, n, 0) \sim [f^n(a)] \quad \textit{and} \quad \mathtt{In}(f, a, n, \mathtt{S}(m)) \sim f^n(a)@\mathtt{In}(f, a, \mathtt{S}(n), m).$$

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

$$\mathtt{in} : N, N \to \alpha$$

is a term in the context $(f : N \to N, a : N)$ *defined by*

$$\mathtt{in} \quad = \quad \lambda x.\mathtt{cond}(\mathtt{base}, \lambda y.\mathtt{ind}),$$

where the base case and the inductive case are

$$\mathtt{base} \quad = \quad [\mathtt{it}(x)] \quad = \quad \mathtt{cons}(\mathtt{it}(x), \mathtt{nil})$$

$$\mathtt{ind} \quad = \quad \mathtt{it}(x)@\mathtt{in}(\mathtt{S}(x))(y) \quad = \quad \mathtt{cons}(\ \mathtt{it}(x),\ \mathtt{in}(\mathtt{S}(x))(y)\ )$$

Proposition 4.3 $\mathtt{In} \in \mathtt{Reg} \cap \mathtt{WTyped} \cap \mathtt{GTC} = \mathtt{CT}\text{-}\lambda$.

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

The proof above includes a type inference from the term `in` to itself. We draw the type inference in the figure 4.3. In this figure, we include a back edge from the † on the top to the † in the bottom, and we marked ♠ the only progress point, which is cyclically repeated in τ. We abbreviate $\Sigma = (\mathtt{nil} : \alpha, \mathtt{cons} : N, \alpha \to N)$ and $\Gamma = \Sigma, (f : N \to N, a : N)$.

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

In conclusion, `In` is some term of CT-λ which can be safely normalized, but which cannot be fully 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 existing β-redexes in one step, also the intermediate steps of the infinite reduction of `In` are not regular.

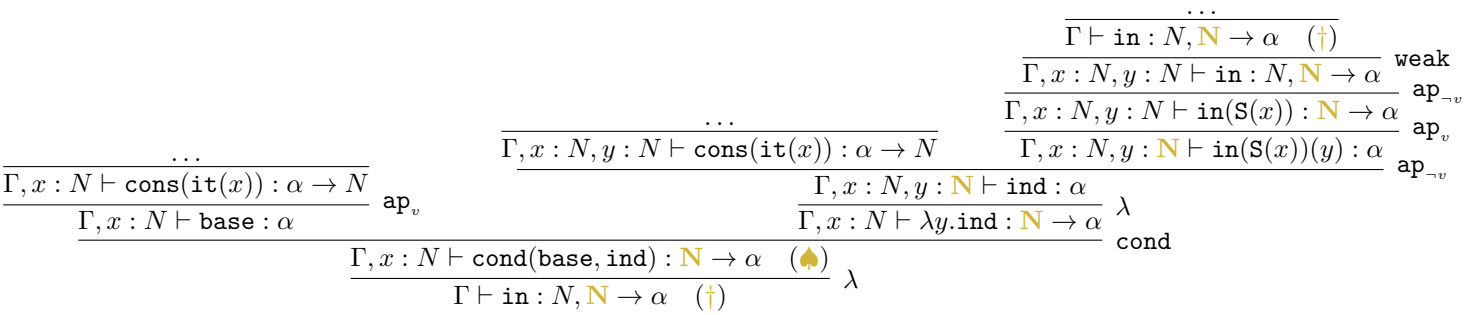

Figure 4.3

5 An example of "interactive use" of CT-λ

Our thesis is that we can have two very similar definitions f_1, f_2 of the same map $F : N, N \to N$ by two regular well-typed terms of $\mathbb{A}$. However, f_1 does not satisfies the Global Trace Condition, therefore $f_1 \notin$ CT-λ is *not* automatically recognized as total, while f_2 satisfies the Global Trace condition and is automatically recognized as total.

Why does it happen? The global trace condition for CT-λ follows the trace of some type-Nvariables, for instance it can follow the trace of the input x^N for $g(x)$, but it cannot follow the trace of a non-variable u used as input for $g(u)$. This means that for the global trace algorithm can follow the trace of the local name x of u in $(\lambda x^N.g(x))(u)$, while cannot follow the trace of u itself in $g(u)$. If x is infinitely decreasing the global trace algorithm can deduce termination in the first case, cannot deduce termination in the second case.

In a sense, the GTC-algorithm is *interactive* for CT-λ. When we believe that $u : N$ is infinitely decreasing in every infinite computation, and that this fact is crucial to insure termination, we should to assign to u a local name x^N. In this way the GTC-algorithm can follow the trace of x^N in any computation. This way of using the global trace algorithm is similar to the way of using the Size-Change-Termination algorithm in Neil Jones.

5.1 The Ackermann Function

The Ackermann function is a good example if we want to test the global trace condition in a case in which it is difficult to prove the convergence of a map. We check that the global trace condition can prove that a definition of Ackermann function in $\mathbb{A}$ is convergent, provided we use local names to denote the sub-expressions ofour definition which are infinitely progressing toward 0.

The Ackermann function $\mathtt{Ack} : N, N \to N$ is a map whose convergence can be proved in**PA**, but only if we use induction over formulas with at least two unbounded quantifiers. For a similar reason, Ack can be defined in system $\mathbb{T}$, but only if we use primitive recursion over terms whose type has degree ≥ 2. Ack grows faster than all primitive recursive maps.

Here is a fixed point definition of Ack. We will express it in CT-λ using a nested conditional.

$$\begin{aligned}\mathtt{Ack}(0, n) &= n + 1\\ \mathtt{Ack}(m + 1, 0) &= \mathtt{Ack}(m, 1)\\ \mathtt{Ack}(m + 1, n + 1) &= \mathtt{Ack}(m, \mathtt{Ack}(m + 1, n))\end{aligned}$$

We can prove that Ack is convergent by induction on the lexicographic order on (m, n). In the next subsections we try to prove convergence of Ack using GTC.

5.2 A Term Representation of Ackermann *without* the Global Trace Condition

We give a first representation $\mathtt{ack1} \in \mathbb{A}$ of the Ackermann function. In this article, we sometimes abbreviate $f(x)(y)$ with $f(x,y)$, in this way we also improve readability.

Definition 5.1 ($\mathtt{ack1}$) *We define* $\mathtt{ack1} \in \mathbb{A}$ *by the following infinite regular term .*

$$\mathtt{ack1} = \lambda \mathbf{N}\mathbf{n}.\mathtt{cond}(\mathtt{S}(\mathbf{n}), \lambda m'.\mathtt{cond}(\mathtt{ack1}(m',1), \lambda n'.\mathtt{ack1}(m', \mathtt{ack1}(\mathtt{S}(m'), n')))(\mathbf{n}))(\mathbf{m})$$

We check that $\mathtt{ack1}$ almost-left-finite. Any infinite path in $\mathtt{ack1}$ must move infinitely many times from $\mathtt{ack1}$ to $\mathtt{ack1}$, and whenever we do so we move to the right-hand-side of a $\mathtt{cond}$. Therefore any infinite path in $\mathtt{ack1}$ is not almost-leftmost, by taking the contrapositive we prove that all almost-leftmost paths in $\mathtt{ack1}$ are finite. $\mathtt{ack1}$ has (unique) type $N, N \to N$.

Checking the fixed point equation for $\mathtt{ack1}$

$$\begin{aligned}
\mathtt{ack1}(0,n) &\mapsto \mathtt{cond}(\mathtt{S}(n), \lambda m'.\mathtt{cond}(\mathtt{ack1}(m',1), \lambda n'.\mathtt{ack1}(m', \mathtt{ack1}(\mathtt{S}(m'), n')))(n))(0) \\
&\mapsto \mathtt{S}(n)
\end{aligned}$$

$$\begin{aligned}
\mathtt{ack1}(\mathtt{S}(m),0) &\mapsto \mathtt{cond}(\mathtt{S}(0), \lambda m'.\mathtt{cond}(\mathtt{ack1}(m',1), \lambda n'.\mathtt{ack1}(m', \mathtt{ack1}(\mathtt{S}(m'), n')))(0))(\mathtt{S}(m)) \\
&\mapsto (\lambda m'.\mathtt{cond}(\mathtt{ack1}(m',1), \lambda n'.\mathtt{ack1}(m', \mathtt{ack1}(\mathtt{S}(m'), n')))(0))(m) \\
&\mapsto \mathtt{cond}(\mathtt{ack1}(m,1), \lambda n'.\mathtt{ack1}(m', \mathtt{ack1}(\mathtt{S}(m'), n')))(0) \\
&\mapsto \mathtt{ack1}(m,1)
\end{aligned}$$

$$\begin{aligned}
\mathtt{ack1}(\underline{\mathtt{S}(m)},\mathtt{S}(n)) &\mapsto \mathtt{cond}(\mathtt{S}(\mathtt{S}(n)), \lambda m'.\mathtt{cond}(\mathtt{ack1}(m',1), \lambda n'.\mathtt{ack1}(m', \mathtt{ack1}(\mathtt{S}(m'), n')))(\mathtt{S}(n)))(\underline{\mathtt{S}(m)}) \\
&\mapsto (\lambda m'.\mathtt{cond}(\mathtt{ack1}(m',1), \lambda n'.\mathtt{ack1}(m', \mathtt{ack1}(\mathtt{S}(m'), n')))(\mathtt{S}(n)))(\underline{m}) \\
&\mapsto \mathtt{cond}(\mathtt{ack1}(m,1), \lambda n'.\mathtt{ack1}(m, \mathtt{ack1}(\mathtt{S}(\underline{m}), n')))(\mathtt{S}(n)) \\
&\mapsto (\lambda n'.\mathtt{ack1}(m, \mathtt{ack1}(\mathtt{S}(\underline{m}), n')))(n) \\
&\mapsto \mathtt{ack1}(m, \mathtt{ack1}(\mathtt{S}(\underline{m}), n))
\end{aligned}$$

The global trace algorithm will fail to prove that $\mathtt{ack1}$ is terminating. The reason is in the last case. The term $\underline{\mathtt{S}(m)}$ of the first line and $\mathtt{S}(\underline{m})$ of the last line are slightly different: $\underline{\mathtt{S}(m)}$ is decomposed into $\underline{m}$ by the $\mathtt{cond}$-reduction, then $\mathtt{S}(\underline{m})$ is constructed by substituting m' of $\mathtt{S}(m')$ by $\underline{m}$. When we loop from the rightmost occurrence of $\mathtt{ack1}$ in the picture above back to the root of $\mathtt{ack1}$, the input $\mathtt{S}(m)$ of $\mathtt{ack1}$ is sent without any change to the first argument of $\mathtt{ack1}$. The globale trace algorithm fails to notice this: in order to notice it, the two terms $\mathtt{S}(m)$ should have the same local name $\mathbf{m}$. As a consequence, the trace of $\mathtt{S}(m)$ is cut in this point and some infinitely progressing trace disappears.

5.3 $\mathtt{ack1} : N \to N \to N$ is not in $\mathtt{GTC}$

Claim $\mathtt{ack1} : N \to N \to N$ is well-typed, but $\mathtt{ack1} \notin \mathtt{GTC}$.

In the following, weakening is implicitly applied.

$$\dfrac{\dfrac{\dfrac{\overset{(\dagger 1)}{\vdash \mathtt{ack1} : \mathbf{N} \to N \to N}}{m' : \mathbf{N} \vdash \mathtt{ack1} : \mathbf{N} \to N \to N}}{m' : \mathbf{N} \vdash \mathtt{ack1}(m') : N \to N}\,(\mathrm{ap_v}) \qquad \dfrac{}{\vdash 1 : N}}{m' : \mathbf{N} \vdash \mathtt{ack1}(m',1) : N}$$

$$\dfrac{\overset{(\dagger 3)}{\vdash \mathtt{ack1} : N \to \mathbf{N} \to N} \qquad \dfrac{\dfrac{}{m' : \mathbf{N} \vdash m' : N}}{m' : \mathbf{N} \vdash \mathtt{S}(m') : N}}{\dfrac{m' : \mathbf{N}, n' : \mathbf{N} \vdash \mathtt{ack1}(\mathtt{S}(m')) : \mathbf{N} \to N}{m' : \mathbf{N}, n' : \mathbf{N} \vdash \mathtt{ack1}(\mathtt{S}(m'), n') : N}\,(\mathrm{ap_v})}\,(\mathrm{ap_{\neg v}})$$

$$\dfrac{\dfrac{\dfrac{\overset{(\dagger 2)}{\vdash \mathtt{ack1} : \mathbf{N} \to N \to N}}{m' : \mathbf{N} \vdash \mathtt{ack1} : \mathbf{N} \to N \to N}}{m' : \mathbf{N} \vdash \mathtt{ack1}(m') : N \to N}\,(\mathrm{ap_v}) \qquad m' : \mathbf{N}, n' : \mathbf{N} \vdash \mathtt{ack1}(\mathtt{S}(m'), n') : N}{\dfrac{m' : \mathbf{N}, n' : \mathbf{N} \vdash \mathtt{ack1}(m', \mathtt{ack1}(\mathtt{S}(m'), n')) : N}{m' : \mathbf{N} \vdash \lambda n'.\mathtt{ack1}(m', \mathtt{ack1}(\mathtt{S}(m'), n')) : \mathbf{N} \to N}}$$

$$\dfrac{m' : \mathbf{N} \vdash \mathtt{ack1}(m',1) : N \qquad m' : \mathbf{N} \vdash \lambda n'.\mathtt{ack1}(m', \mathtt{ack1}(\mathtt{S}(m'), n')) : \mathbf{N} \to N}{\dfrac{\dfrac{n : \mathbf{N}, m' : \mathbf{N} \vdash \mathtt{cond}(\mathtt{ack1}(m',1), \lambda n'.\mathtt{ack1}(m', \mathtt{ack1}(\mathtt{S}(m'), n'))) : \mathbf{N} \to N}{n : \mathbf{N}, m' : \mathbf{N} \vdash \mathtt{cond}(\mathtt{ack1}(m',1), \lambda n'.\mathtt{ack1}(m', \mathtt{ack1}(\mathtt{S}(m'), n')))(n) : N}\,(\mathrm{ap_v})}{n : \mathbf{N} \vdash \lambda m'.\mathtt{cond}(\mathtt{ack1}(m',1), \lambda n'.\mathtt{ack1}(m', \mathtt{ack1}(\mathtt{S}(m'), n')))(n) : \mathbf{N} \to N}}\,(\mathrm{cond})$$

$$\dfrac{\dfrac{\dfrac{}{n : \mathbf{N} \vdash n : N}}{n : \mathbf{N} \vdash \mathtt{S}(n) : N} \qquad n : \mathbf{N} \vdash \lambda m'.\mathtt{cond}(\mathtt{ack1}(m',1), \lambda n'.\mathtt{ack1}(m', \mathtt{ack1}(\mathtt{S}(m'), n')))(n) : \mathbf{N} \to N}{\dfrac{\dfrac{m : \mathbf{N}, n : \mathbf{N} \vdash \mathtt{cond}(\mathtt{S}(n), \lambda m'.\mathtt{cond}(\mathtt{ack1}(m',1), \lambda n'.\mathtt{ack1}(m', \mathtt{ack1}(\mathtt{S}(m'), n')))(n)) : \mathbf{N} \to N}{m : \mathbf{N}, n : \mathbf{N} \vdash \mathtt{cond}(\mathtt{S}(n), \lambda m'.\mathtt{cond}(\mathtt{ack1}(m',1), \lambda n'.\mathtt{ack1}(m', \mathtt{ack1}(\mathtt{S}(m'), n')))(n))(m) : N}\,(\mathrm{ap_v})}{\vdash \mathtt{ack1} : \mathbf{N} \to \mathbf{N} \to N \ (\dagger)}}\,(\mathrm{cond})$$

The problem is with the rightmost rule $(\mathrm{ap}_{\neg\mathrm{v}})$ in the picture above. The same value $\mathtt{S}(m') = m$ is send to the first argument of `ack1`, but this fact is not reported in the trace. This cuts the trace chasing of **N** (red color). A similar situation arises with the leftmost rule $(\mathrm{ap}_{\neg\mathrm{v}})$ in the picture above: the trace chasing of **N** (gold color) is cut.

As a consequence, the infinite path $(\dagger) \rightsquigarrow (\dagger 1) \rightsquigarrow (\dagger) \rightsquigarrow (\dagger 3) \rightsquigarrow (\dagger) \rightsquigarrow (\dagger 1) \rightsquigarrow (\dagger) \rightsquigarrow (\dagger 3) \rightsquigarrow \cdots$ does not contain any infinite trace, and with more reason any infinitely progressing trace.

5.4 Second term representation: `ack2`

We give the second representation `ack2` of Ackermann function. In this case we inform the global trace algorithm that the local term $\mathtt{S}(m')$ is equal to m in any computation by writing m in the place of $\mathtt{S}(m')$.

Definition 5.2 (`ack2`) *We define* `ack2` *by the following infinite regular term of* $\mathbb{A}$.

$$\mathtt{ack2} = \lambda \mathbf{mn}.\mathtt{cond}(\mathtt{S}(\mathbf{n}), \lambda m'.\mathtt{cond}(\mathtt{ack2}(m',1), \lambda n'.\mathtt{ack2}(m', \mathtt{ack2}(\mathbf{m}, n')))(\mathbf{n}))(\mathbf{m})$$

Again, the term `ack2` is almost-left-finite and has type $N, N \to N$.

Checking the fixed point equation for `ack2`

We check only the last case.

$$\begin{aligned}
\mathtt{ack2}(\underline{\mathtt{S}(m)}, \mathtt{S}(n)) &\mapsto \mathtt{cond}(\mathtt{S}(\mathtt{S}(n)), \lambda m'.\mathtt{cond}(\mathtt{ack2}(m',1), \lambda n'.\mathtt{ack2}(m', \mathtt{ack2}(\underline{\mathtt{S}(m)}, n')))(\mathtt{S}(n)))(\underline{\mathtt{S}(m)}) \\
&\mapsto (\lambda m'.\mathtt{cond}(\mathtt{ack2}(m',1), \lambda n'.\mathtt{ack2}(m', \mathtt{ack2}(\underline{\mathtt{S}(m)}, n')))(\mathtt{S}(n)))(\underline{m}) \\
&\mapsto \mathtt{cond}(\mathtt{ack2}(m,1), \lambda n'.\mathtt{ack2}(m, \mathtt{ack2}(\underline{\mathtt{S}(m)}, n')))(\mathtt{S}(n)) \\
&\mapsto (\lambda n'.\mathtt{ack2}(m, \mathtt{ack2}(\underline{\mathtt{S}(m)}, n')))(n) \\
&\mapsto \mathtt{ack2}(m, \mathtt{ack2}(\underline{\mathtt{S}(m)}, n))
\end{aligned}$$

The point of `ack2` is that the global trace algorithm by checking the traces discover that $\underline{\mathtt{S}(m)}$ at the first line and the one at the last line are exactly the same, thanks to the fact that they have the same local name m.

5.5 $\mathtt{ack2} : N \to N \to N$ is in `GTC`

Proposition 5.3 $\mathtt{ack2} : N \to N \to N$ *has a proof* Π *that satisfies* `GTC`.

Proof Below is a proof Π of $\mathtt{ack2} : N \to N \to N$. In the following, weakening is implicitly applied.

$$\dfrac{\dfrac{\dfrac{\dfrac{n : \mathbf{N} \vdash n : N}{n : \mathbf{N} \vdash \mathtt{S}(n) : N} \quad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\overset{(\dagger 1)}{\vdash \mathtt{ack2} : \mathbf{N} \to N \to N}}{m' : \mathbf{N} \vdash \mathtt{ack2} : \mathbf{N} \to N \to N}}{m' : \mathbf{N} \vdash \mathtt{ack2}(m') : N \to N}(\mathrm{ap_v}) \quad \vdash 1 : N}{m' : \mathbf{N} \vdash \mathtt{ack2}(m',1) : N} \quad \dfrac{\dfrac{\dfrac{\dfrac{\overset{(\dagger 2)}{\vdash \mathtt{ack2} : \mathbf{N} \to N \to N}}{m' : \mathbf{N} \vdash \mathtt{ack2} : \mathbf{N} \to N \to N}}{m' : \mathbf{N} \vdash \mathtt{ack2}(m') : N \to N}(\mathrm{ap_v}) \quad \dfrac{\dfrac{\dfrac{\dfrac{\overset{(\dagger 3)}{\vdash \mathtt{ack2} : \mathbf{N} \to \mathbf{N} \to N}}{m : \mathbf{N} \vdash \mathtt{ack2} : \mathbf{N} \to \mathbf{N} \to N}}{m : \mathbf{N} \vdash \mathtt{ack2}(m) : \mathbf{N} \to N}(\mathrm{ap_v})}{m : \mathbf{N}, n' : \mathbf{N} \vdash \mathtt{ack2}(m) : \mathbf{N} \to N}}{m : \mathbf{N}, n' : \mathbf{N} \vdash \mathtt{ack2}(m,n') : N}(\mathrm{ap_v})}{m : \mathbf{N}, m' : \mathbf{N}, n' : \mathbf{N} \vdash \mathtt{ack2}(m', \mathtt{ack2}(m,n')) : N}}{m : \mathbf{N}, m' : \mathbf{N} \vdash \lambda n'.\mathtt{ack2}(m', \mathtt{ack2}(m,n')) : \mathbf{N} \to N}}{m : \mathbf{N}, n : \mathbf{N}, m' : \mathbf{N} \vdash \mathtt{cond}(\mathtt{ack2}(m',1), \lambda n'.\mathtt{ack2}(m', \mathtt{ack2}(m,n'))) : \mathbf{N} \to N}(\mathrm{cond})}{m : \mathbf{N}, n : \mathbf{N}, m' : \mathbf{N} \vdash \mathtt{cond}(\mathtt{ack2}(m',1), \lambda n'.\mathtt{ack2}(m', \mathtt{ack2}(m,n')))(n) : N}(\mathrm{ap_v})}{m : \mathbf{N}, n : \mathbf{N} \vdash \lambda m'.\mathtt{cond}(\mathtt{ack2}(m',1), \lambda n'.\mathtt{ack2}(m', \mathtt{ack2}(m,n')))(n) : \mathbf{N} \to N}}{m : \mathbf{N}, n : \mathbf{N} \vdash \mathtt{cond}(\mathtt{S}(n), \lambda m'.\mathtt{cond}(\mathtt{ack2}(m',1), \lambda n'.\mathtt{ack2}(m', \mathtt{ack2}(m,n')))(n)) : \mathbf{N} \to N}(\mathrm{cond})}{m : \mathbf{N}, n : \mathbf{N} \vdash \mathtt{cond}(\mathtt{S}(n), \lambda m'.\mathtt{cond}(\mathtt{ack2}(m',1), \lambda n'.\mathtt{ack2}(m', \mathtt{ack2}(m,n')))(n))(m) : N}(\mathrm{ap_v})}{\vdash \mathtt{ack2} : \mathbf{N} \to \mathbf{N} \to N \ (\dagger)}$$

We check that this proof Π satisfies the global trace condition. We first note that:

- The path $(\dagger) \rightsquigarrow (\dagger 1)$ contains a progressing blue trace $\tau_1 = (\mathbf{N}, \mathbf{N}, \mathbf{N}, \ldots, \mathbf{N})$. The gold trace is cut in $(\dagger 1)$.

- The path (†) $\rightsquigarrow$ (†2) contains a progressing blue trace $\tau_2 = (\mathbf{N}, \mathbf{N}, \mathbf{N}, \ldots, \mathbf{N})$. The gold trace is cut in (†2).
- The path (†) $\rightsquigarrow$ (†3) contains a progressing gold trace $\tau_3' = (\mathbf{N}, \ldots, \mathbf{N})$ and a non-progressing red trace $\tau_3 = (\mathbf{N}, \mathbf{N}, \mathbf{N}, \ldots, \mathbf{N})$.

Each trace τ_i, blue or red, can be composed with each trace τ_j.

The first steps of each τ_i are in common and we write them with the blue and red colors superposed. Then the blue and the red traces fork.

Take an infinite path π from this proof Π in order to prove that it includes some infinitely progressing trace.

1. If π passes through (†1) or (†2) infinitely many times, we take its infinitely progressing trace by combining τ_1, τ_2, and τ_3, according if we pass through(†1) or (†2) or (†3).
2. If π passes through (†1) and (†2) only finitely many times, namely it eventually becomes a loop of (†) and (†3), from this point on we take its infinitely progressing trace by repeating τ_3' for infinitely many times.

6 Subject Reduction for Well-Typed Infinite Lambda Terms

We show the subject reduction for well-typed terms of $\mathbb{\Lambda}$, and also show the global trace condition is preserved by reductions. We first introduce some auxiliary notations for proofs of them.

Let X be a set of variables of the form x^T. We write Γ_X for the sublist of Γ consisting of all $x^T : T$ such that $x^T \in X$. Let Γ and Δ be contexts of $\mathbb{\Lambda}$. The merged context $\Gamma\sharp\Delta$ is defined by $\Gamma\star(\Delta_{\mathrm{FV}(\Delta)\setminus\mathrm{FV}(\Gamma)})$.

Let θ be a renaming, and $S_1 = \Gamma_1 \vdash t_1 : \vec{B_1} \to N$ and $S_2 = \Gamma_2[\theta] \vdash t_2[\theta] : \vec{B_2} \to N$ be sequents. Let k_1 and k_2 be indexes of N-arguments of t_1 and t_2, respectively. Then we write $(k_1, S_1) \sim^{\theta}_{\mathtt{idx}} (k_2, S_2)$ (or $(k_1, t_1) \sim^{\theta}_{\mathtt{idx}} (k_2, t_2)$ for short) if they are respectively indexes of named arguments for some y and $\theta(y)$, or are respectively those of unnamed arguments at the same position i in $\vec{B_1}$ and $\vec{B_2}$, namely $k_1 = |\Gamma_1| + i$ and $k_2 = |\Gamma_2| + i$, where $|\Gamma_1|$ and $|\Gamma_2|$ are the sizes of Γ_1 and Γ_2. Note that the index equivalent to k_1 is unique (if it exists), namely $(k_1, t_1) \sim^{\theta}_{\mathtt{idx}} (k_2, t_2)$ and $(k_1, t_1) \sim^{\theta}_{\mathtt{idx}} (k'_2, t_2)$ implies $k_2 = k'_2$. We write $(k_1, t_1) \sim_{\mathtt{idx}} (k_2, t_2)$ if θ is the identity renaming.

In this paper we do not implicitly identify α-equivalent terms, in order to make simpler to state the global trace condition. For this reason, proofs of this section requires some delicate treatment of variables.

We consider the following restricted λ rule (called λ') as follows:

- λ'-rule. If $\Gamma, x^A : A \vdash b : B$ and $\mathrm{FV}(\Gamma) = \mathrm{FV}(\lambda x.b)$, then $\Gamma \vdash \lambda x^A.b : A \to B$.

The next lemma says that if $\Gamma \vdash t : A$ is provable, then $t : A$ can be shown in a restricted context $\Gamma_{\mathrm{FV}(t)}$ even if the rule λ is restricted to λ'.

Lemma 6.1 *Assume $\Pi : \Gamma \vdash t : A$. Then $\Pi' : \Gamma_{\mathrm{FV}(t)} \vdash t : A$ for some Π' that may have instances of the rule λ' and not have those of the rule λ. Moreover if the global trace condition holds for Π, then it also holds for Π'.*

Proof First we call the following admissible rule $\lambda'\mathtt{weak}$:

- $\lambda'\mathtt{weak}$-rule. If $\Gamma, x^A : A \vdash b : B$, $\mathrm{FV}(\Gamma) = \mathrm{FV}(\lambda x.b)$ and $\Gamma \subsetneq \Gamma'$, then $\Gamma' \vdash \lambda x^A.b : A \to B$.

We write $\mathtt{Rule}'$ as the set of rule instances obtained by removing those of λ from $\mathtt{Rule}$ and adding those of $\lambda'\mathtt{weak}$. Note that if we have a proof of a sequent with $\lambda'\mathtt{weak}$, then we also have a proof of the same sequent not with $\lambda'\mathtt{weak}$ by a proof transformation that splits each $\lambda'\mathtt{weak}$ by λ' and $\mathtt{weak}$. Also note that this proof transformation preserves the global trace condition.

Let Π be (T, ϕ). For each $l \in T$, we write $\Gamma_l \vdash t_l : A_l$ for the conclusion of $\phi(l) \in \mathtt{Rule}$. To show the lemma, from a given proof Π, it is enough to construct a proof Π' of $\Gamma_{\mathrm{FV}(t)} \vdash t : A$ with $\mathtt{Rule}'$.

We define the set of nodes of Π' is T, which is the same one of Π. For each $l \in T$, we define $\phi'(l)$ and Γ'_l that satisfies the following requirements:

(a) $\Gamma'_l \vdash t_l : A_l$ is the conclusion of $\phi'(l) \in \mathtt{Rule}'$,
(b) $(\Gamma_l)_{\mathrm{FV}(t_l)} \subsetneq \Gamma'_l \subsetneq \Gamma_l$ and $\Gamma'_{\mathtt{nil}} = \Gamma_{\mathrm{FV}(t)}$,
(c) if $\tilde{k}, t_{l\star(i)}$ is the successor of k, t_l in Π and k is an index of some unname argument or a named argument of some name $z \in \mathrm{FV}(t_l)$, then there are $\tilde{k'}$ and k' such that $\tilde{k'}, t_{l\star(i)}$ is the successor of k', t_l in Π', $(k, t_l) \sim_{\mathtt{idx}} (k', t_l)$, and $(\tilde{k}, t_{l\star(i)}) \sim_{\mathtt{idx}} (\tilde{k'}, t_{l\star(i)})$. Moreover, if k is an index of a progressing argument, then so is k'.

Note that the proof is done if we construct Π' that satisfies these requirements.

We define $\Gamma'_{\mathtt{nil}} = \Gamma_{\mathrm{FV}(t)}$. Then it satisfies (b) since $t_{\mathtt{nil}} = t$. Next, assuming the induction hypothesis that Γ'_l which satisfies (b) is already defined, we define $\phi'(l)$ and $\Gamma_{l\star(i)}$, for each i such that $l\star(i) \in T$, that satisfies (a), (b), and (c). It is done by the case analysis of $\phi(l)$.

The case of $\phi(l) = \Gamma_l \vdash x : A_l$, which is an instance of the rule $\mathtt{var}$ with $x : A_l \in \Gamma_l$. Then define $\phi'(l) = \Gamma'_l \vdash x : A_l$. This is an instance of $\mathtt{var}$ because $x : A_l \in (\Gamma_l)_{\mathrm{FV}(x)} \subsetneq \Gamma'_l$ by (b).

The case of $\phi(l) = (\Gamma_{l\star(1)} \vdash t_l : A_l, \Gamma_l \vdash t_l : A_l)$, which is an instance of the rule $\mathtt{weak}$ with $t_{l\star(1)} = t_l$, $A_{l\star(1)} = A_l$, and $\Gamma_{l\star(1)} \subsetneq \Gamma_l$. Let ψ be the unique map that determines $\Gamma_{l\star(1)} \subsetneq \Gamma_l$. Then define $\Gamma'_{l\star(1)}$ such that $\Gamma'_{l\star(1)} \subsetneq \Gamma'_l$ determined by the induced map from ψ restricting the range to $\mathrm{FV}(\Gamma'_l)$. Note that $\mathrm{FV}(\Gamma'_{l\star(1)}) = \mathrm{FV}(\Gamma'_l) \cap \mathrm{FV}(\Gamma_{l\star(1)})$. Then it satisfies (b) since $\mathrm{FV}(t_l) \subseteq \mathrm{FV}(\Gamma'_l) \cap \mathrm{FV}(\Gamma_{l\star(1)}) = \mathrm{FV}(\Gamma'_{l\star(1)}) \subseteq \mathrm{FV}(\Gamma_{l\star(1)})$ by (b) for Γ'_l. Define $\phi'(l) = (\Gamma'_{l\star(1)} \vdash t_l : A_l, \Gamma'_l \vdash t_l : A_l)$ as an instance of the

rule $\mathtt{weak}$. Hence the requirement (a) holds. We can also show (c): if k is an index in $\Gamma_l \vdash t_l : A_l$ for a name $z \in \mathrm{FV}(t_{l\star(1)}) = \mathrm{FV}(t_l)$, then we can take an index k' in $\Gamma'_l \vdash t_l : A_l$ for z by $\mathrm{FV}(t_l) \subseteq \mathrm{FV}(\Gamma'_l)$ by (b). An index $\tilde{k'}$ for the name z can be taken from $\Gamma'_{l\star(1)} \vdash t_l : A_l$ since $z \in \mathrm{FV}(t_l) \subseteq \mathrm{FV}(\Gamma'_{l\star(1)})$.

The case of $\phi(l) = (\Gamma_l, z : C \vdash b : \vec{B} \to N, \Gamma_l \vdash \lambda z^C.b : C, \vec{B} \to N)$ that is an instance of the rule λ. Define $\Gamma'_{l\star(1)} = (\Gamma_l)_{\mathrm{FV}(\lambda z.b)}, z : C$ and $\phi'(l) = ((\Gamma_l)_{\mathrm{FV}(\lambda z.b)}, z : C \vdash b : \vec{B} \to N, \Gamma'_l \vdash \lambda z.b : C, \vec{B} \to N)$. This is an instance of the rule $\lambda'\mathtt{weak}$, since $(\Gamma_l)_{\mathrm{FV}(\lambda z.b)} \subsetneq \Gamma'_l$ by the induction hypothesis. Trivially we have (a). We also have (b) by $(\Gamma_l, z : C)_{\mathrm{FV}(b)} \subsetneq ((\Gamma_l)_{\mathrm{FV}(\lambda z.b)}, z : C) \subsetneq (\Gamma_l, z : C)$. The requirement (c) holds: if k is an index of some named argument $y \in \mathrm{FV}(\lambda z.b)$ in Γ_l, then k' can be taken as the index of y in Γ'_l by (b) for Γ_l. If k is an index of some unnamed argument in $C, \vec{B}$, then k' can be taken as the index of some unnamed argument. In both cases, their successors $\tilde{k}$ and $\tilde{k'}$ are uniquely determined by k and k', respectively.

The case that $\phi(l)$ is an instance of the rule $\mathtt{ap}_v$ whose conclusion is $\Gamma_l \vdash f(x^B) : \vec{C} \to N$ with $x : B \in \Gamma_l$. Define $\Gamma'_{l\star(1)} = \Gamma'_l$. This satisfies (b) by the induction hypothesis. Then define $\phi'(l) = (\Gamma'_{l\star(1)} \vdash f : B, \vec{C} \to N, \Gamma'_l \vdash f(x^B) : \vec{C} \to N)$ as an instance of the rule $\mathtt{ap}_v$. This satisfies (a). In order to show (c), take an index k for $\Gamma_l \vdash f(x^B) : \vec{C} \to N$, which is the one for a named argument $y \in \mathrm{FV}(f(x^B))$ in Γ_l or for some unnamed argument of $\vec{C}$. For the latter case, the indexes $\tilde{k}$, k' and $\tilde{k'}$ can be taken as those of the unnamed argument in $\vec{C}$ at the same position as k. For the former case, we take k' as the index for y in Γ'_l. In order to take $\tilde{k'}$, we further have two subcases according to $\tilde{k}$: The first one is $\tilde{k}$ is the index for the same named argument as k, and the second one is $\tilde{k}$ is the index for the unnamed argument, namely $B = N$. In both subcases, we can take the equivalent $\tilde{k'}$ to $\tilde{k}$ as we wished.

The case that $\phi(l)$ is an instance of the rule $\mathtt{ap}_{\neg v}$ whose conclusion is $\Gamma_l \vdash f(b^B) : A_l$. Define $\Gamma'_{l\star(1)} = \Gamma'_{l\star(2)} = \Gamma'_l$. This satisfies (b) by the induction hypothesis. Then define $\phi'(l) = (S_1, S_2, \Gamma'_l \vdash f(b^B) : \vec{C} \to N)$, where S_1 is $\Gamma'_l \vdash f : B, \vec{C} \to N$ and S_2 is $\Gamma'_l \vdash b : B$, as an instance of $\mathtt{ap}_{\neg v}$. This satisfies (a). In order to show (c), take an index k for $\Gamma_l \vdash f(x^B) : \vec{C} \to N$, which is the one for a named argument $y \in \mathrm{FV}(f(x^B))$ in Γ_l or for some unnamed argument of $\vec{C}$. For the former case, the successor $\tilde{k}$ of k is uniquely taken. By $\mathrm{FV}(f(b)) \subseteq \mathrm{FV}(\Gamma'_l)$, the index k' equivalent to k is uniquely taken. Then the successor $\tilde{k'}$ of k' is also taken as we wished. For the latter case, the successor $\tilde{k}$ of k is uniquely taken as the one for unnamed argument at the same position as k. Then k' equivalent to k is uniquely taken, and its successor $\tilde{k'}$ is also taken as we wished.

The case that $\phi(l)$ is an instance of the rule 0 whose conclusion is $\Gamma_l \vdash 0 : N$. Define $\phi'(l) = \Gamma'_l \vdash 0 : N$ as an instance of the rule 0. This satisfies the requirements (a), (b), and (c).

The case that $\phi(l)$ is an instance of the rule $\mathtt{S}$ whose conclusion is $\Gamma_l \vdash \mathtt{S}(t_{l\star(1)}) : N$ with $A_l = N$. Define $\Gamma'_{l\star(1)} = \Gamma'_l$, and $\phi'(l) = (\Gamma'_l \vdash t_{l\star(1)} : N, \Gamma'_l \vdash \mathtt{S}(t_{l\star(1)}) : N)$ as an instance of $\mathtt{S}$. They satisfy (a) and (b). The requirement (c) is also satisfied: Take an index k for $\Gamma_l \vdash \mathtt{S}(t_{l\star(1)}) : N$, which is the one for a named argument $y \in \mathrm{FV}(\mathtt{S}(t_{l\star(1)})$ in Γ_l. Then the successor $\tilde{k}$ of k is uniquely taken. By $\mathrm{FV}(t_{l\star(1)}) \subseteq \mathrm{FV}(\Gamma'_l)$, the index k' equivalent to k is uniquely taken. Then the successor $\tilde{k'}$ of k' is also taken as we wished.

The case that $\phi(l)$ is an instance of the rule $\mathtt{cond}$ whose conclusion is $\Gamma_l \vdash \mathtt{cond}(f, g) : N, \vec{C} \to N$. Define $\Gamma'_{l\star(1)} = \Gamma'_{l\star(2)} = \Gamma'_l$, and $\phi'(l) = (S_1, S_2, \Gamma'_l \vdash \mathtt{cond}(f, g) : N, \vec{C} \to N)$, where S_1 is $\Gamma'_l \vdash f : \vec{C} \to N$ and S_2 is $\Gamma'_l \vdash g : N, \vec{C} \to N$, as an instance of $\mathtt{cond}$. They satisfy (a) and (b). To show (c), take an index k for $\Gamma_l \vdash \mathtt{cond}(f, g) : N, \vec{C} \to N$ as required. We need to consider three cases: k is the index for a named argument $y \in \mathrm{FV}(\mathtt{cond}(f, g))$ in Γ_l, is the one for an unnamed argument in $\vec{C}$, or is the one for an unnamed argument in N (the first one of $N, \vec{C} \to N$). For the first and second cases, we can take $\tilde{k}$, k', and $\tilde{k'}$ similar to the other cases. For the last case, the successor $\tilde{k}$ of k should be taken as the index of the unnamed N-argument of $\Gamma'_l \vdash g : N, \vec{C} \to N$ at the same position as k. Then k' and $\tilde{k'}$ are taken as those equivalent to k and $\tilde{k}$, respectively. Note that this k is an index of a progressing argument N, and so is k'.

Lemma 6.2 (Substitution lemma) *Assume that $\Pi_u : \Delta \vdash u : A$ and $\Pi_t : \Gamma, x : A \vdash t : B$ hold. Then there exists Π° such that $\Pi^\circ : \Delta \sharp \Gamma \vdash t[u/x] : B$. Moreover, if both Π_u and Π_t satisfy the global trace condition, then Π° also satisfies it.*

Proof Assume that $\Pi_u : \Delta \vdash u : A$ and $\Pi_t : \Gamma, x : A \vdash t : B$ hold. By Lemma 6.1, without loss of

generality, we can assume $\mathrm{FV}(\Delta) = \mathrm{FV}(u)$ and Π_t is a proof with the restricted λ rule (the λ' rule). Let $\Pi_u = (T_u, \phi_u)$ and $\Pi_t = (T_t, \phi_t)$. For each $l \in T_t$, we write $\Gamma_l \vdash t_l : C_l$ for the conclusion of $\phi_t(l)$. We use $\boldsymbol{x}$ to mark occurences of x in Π_t that connects with the explicit $\boldsymbol{x}$ of $\Gamma, \boldsymbol{x} : A \vdash t : B$. In the following, we construct a proof $\Pi^\circ = (T^\circ, \phi^\circ)$ of $\Delta \sharp \Gamma \vdash t[u/x] : B$ such that

(a1) $T^\circ = T_t \cup T_{\mathtt{var}} \cup T_{\mathtt{ap}_v}$, where $T_{\mathtt{var}} = \{l\star(1)\star l' \mid l \in T_t,\ \phi(l) = \mathtt{var}$ of $\boldsymbol{x}$, and $l' \in T_u\}$ and $T_{\mathtt{ap}_v} = \{l\star(2)\star l' \mid u$ is not a variable, $l \in T_t,\ \phi(l) = \mathtt{ap}_v$ of $f(\boldsymbol{x})$ for some f, and $l' \in T_u\}$. The explicit 1 of $l\star(1)\star l' \in T_{\mathtt{var}}$ is called the switching point of $l\star(1)\star l'$. The explicit 2 of $l\star(2)\star l' \in T_{\mathtt{ap}_v}$ is also called the switching point of $l\star(2)\star l'$.

(a2) For each $l\star(i)\star l' \in T_{\mathtt{var}} \cup T_{\mathtt{ap}_v}$ with switching point i, we have $\phi^\circ(l\star(i)\star l') = \phi_u(l')$.

Moreover, for all $l \in T_t$, the rule instance $\phi^\circ(l)$ satisfies the following requirements with an auxiliary function $\sigma^\circ : T_t \to \mathtt{Seq}$ and a substitution θ_l:

(b1) The sequent $\sigma^\circ(l)$ has the form $\Gamma_l^\circ \vdash t_l[\theta_l] : C_l$. The substitution θ_l is $\{u/\boldsymbol{x}\} \cup \theta_{\mathrm{ren}}$ if $\boldsymbol{x} : A \in \Gamma_l$, and is θ_{ren} otherwise, where θ_{ren} is some renaming. $\Gamma_l^\circ \sim \Delta_l \sharp (\Gamma_l)_{\overline{\boldsymbol{x}}}[\theta_l]$ holds, where Δ_l is Δ if $\boldsymbol{x} : A \in \Gamma_l$, and is $\emptyset$ otherwise.

(b2) $\sigma^\circ(l)$ is the conclusion of $\phi^\circ(l)$, and $\sigma^\circ(l\star(i))$ is the i-th assumption of $\phi^\circ(l)$ if $l\star(i) \in T_t$.

(b3) Assume $\tilde{k}, t_{l\star(i)}$ is a successor of k, t_l and k is not a named index for $\boldsymbol{x}$. Then there exist $\tilde{k^\circ}$ and k° such that $\tilde{k^\circ}, t_{l\star(i)}[\theta_{l\star(i)}]$ is a successor of $k^\circ, t_l[\theta_l]$ and $(k, t_l) \sim_{\mathtt{idx}}^{(\theta_l)_{\overline{\boldsymbol{x}}}} (k^\circ, t_l[\theta_l])$. If k is an index of a progressing argument, then so is k°.

Note that if we have Π° that satisfies the requirements, its possibly infinite path is a path of Π_t or a path of Π_u or a path $l\star(i)\star l'$, where l is a path of T_t, i is a switching point, and l' is a path of T_u. Hence Π° is Almost-left-finite, since Π_t and Π_u are Almost-left-finite. In addition, if Π_t and Π_u satisfies the global trace condition, so is Π° by (b3). Therefore, to complete the proof, it is enough to construct Π°.

First, for each $l\star(j)\star l' \in T_{\mathtt{var}} \cup T_{\mathtt{ap}_v}$ with a switching point j, we define $\phi^\circ(l\star(j)\star l') = \phi_u(l')$.

In the following, for each $l \in T_t$, we inductively define Γ_l°, $\phi^\circ(l)$, and $\sigma(l)$ that satisty (b1), (b2), and (b3). The case of $l = \mathtt{nil}$, define $\Gamma^\circ_{\mathtt{nil}} = \Delta \sharp \Gamma$ and $\theta_{\mathtt{nil}} = \{u/\boldsymbol{x}\}$, and define $\sigma^\circ(\mathtt{nil})$ as $\Delta \sharp \Gamma \vdash t[u/\boldsymbol{x}] : A$. Hence we have (b1) since $\Gamma^\circ_{\mathtt{nil}} = \Delta \sharp \Gamma \sim \Delta \sharp (\Gamma, \boldsymbol{x} : A)_{\overline{\boldsymbol{x}}}[\theta_{\mathtt{nil}}]$.

For each $l \in T_t$, with the induction hypothesis that $\sigma^\circ(l)$ and θ_l that satisfy (b1) are already defined, we define $\phi^\circ(l)$, and define $\theta_{l\star(i)}$ and $\sigma^\circ(l\star(i))$ when $l\star(i) \in T_t$, such that they satisfy (b1), (b2), and (b3). We perform this by the case analysis of $\phi(l)$.

The case $\phi(l) = (\Gamma_l \vdash \boldsymbol{x} : A)$ that is an instance of $\mathtt{var}$. We can define $\phi^\circ(l) = (\Delta \vdash u : A, \Gamma_l^\circ \vdash u : A)$ as $\mathtt{weak}$, since $\Delta \subsetneq \Delta \sharp (\Gamma_l)_{\overline{\boldsymbol{x}}}[\theta_l] \sim \Gamma_l^\circ$ holds by (b1). As $l\star(i) \notin T_t$, (b1), (b2), and (b3) trivially hold.

The case $\phi(l) = (\Gamma_l \vdash y : B)$ that is an instance of $\mathtt{var}$ with $y \neq \boldsymbol{x}$. We can define $\phi^\circ(l) = (\Gamma_l^\circ \vdash \theta_l(y) : B)$ as $\mathtt{var}$, since $\theta_l(y) : B \in (\Gamma_l)_{\overline{\boldsymbol{x}}}[\theta_l] \subsetneq \Delta_l \sharp (\Gamma_l)_{\overline{\boldsymbol{x}}}[\theta_l] \sim \Gamma_l^\circ$ holds by (b1). As $l\star(i) \notin T_t$, (b1), (b2), and (b3) trivially hold.

The case $\phi(l) = (S_1, S_2, \Gamma_l \vdash f(b) : C)$ that is an instance of $\mathtt{ap}_{\neg v}$, where $S_1 = \Gamma_l \vdash f : B \to C$, $S_2 = \Gamma_l \vdash b : B$, and b is not a variable. Define $\phi^\circ(l) = (S_1', S_2', \Gamma_l^\circ \vdash f[\theta_l](b[\theta_l]) : C)$ as $\mathtt{ap}_{\neg v}$, where $S_1' = \Gamma_l^\circ \vdash f[\theta_l] : B \to C$ and $S_2' = \Gamma_l^\circ \vdash b[\theta_l] : B$, since $b[\theta_l]$ is not a variable. Also define $\sigma^\circ(l\star(1)) = S_1'$, $\sigma^\circ(l\star(2)) = S_2'$, and $\theta_{l\star(1)} = \theta_{l\star(2)} = \theta_l$. Then (b1) and (b2) trivially hold. To check (b3), assume that $\tilde{k}, t$ is a successor of $k, f(b)$, where t is f or b, and k is not a named index for $\boldsymbol{x}$. Then (i) both k and $\tilde{k}$ are named indexes for some variable $y^D \in \mathrm{Dom}(\Gamma_l)$, (ii) both k and $\tilde{k}$ are unnamed indexes in C. For the case (ii), take k' and $\tilde{k'}$ as the same indexes as k and $\tilde{k}$, respectively. For the case (i), we have $\theta_l(y) : D \in (\Gamma_l)_{\overline{\boldsymbol{x}}}[\theta_l] \subsetneq \Gamma_l^\circ \subsetneq \Gamma^\circ_{l\star(1)}$ by $y \neq \boldsymbol{x}$. Then take k' and $\tilde{k'}$ as named indexes for $\theta_l(y)$. In each case, k' and $\tilde{k'}$ satisfy (b3) as expected.

The case $\phi(l) = (S_1, \Gamma_l \vdash f(\boldsymbol{x}) : C)$ that is an instance of $\mathtt{ap}_v$, where $S_1 = \Gamma_l \vdash f : A \to C$ and $\boldsymbol{x} : A \in \Gamma_l$. We need to consider the two subcases whether u is a variable or not.

- If u is a variable y^B, we can define $\phi^\circ(l) = (S_1', \Gamma_l^\circ \vdash f[\theta_l](y) : C)$, where $S_1' = \Gamma_l^\circ \vdash f[\theta_l] : A \to C$ as $\mathtt{ap}_v$, since $y : B \in \Delta \subsetneq \Delta \sharp (\Gamma_l)_{\overline{\boldsymbol{x}}}[\theta_l] \sim \Gamma_l^\circ$ holds by (b1). Also define $\sigma^\circ(l\star(1)) = S_1'$ and $\theta_{l\star(1)} = \theta_l$. Then (b1) and (b2) trivially hold. (b3) is checked in a similar way to the case $\mathtt{ap}_{\neg v}$.
- If u is not a variable, define $\phi^\circ(l) = (S_1', S_2', \Gamma_l^\circ \vdash f[\theta_l](u) : C)$, where $S_1' = \Gamma_l^\circ \vdash f[\theta_l] : A \to C$ and $S_2' = \Gamma_l^\circ \vdash u : A$ as $\mathtt{ap}_{\neg v}$. Also define $\sigma^\circ(l\star(1)) = S_1'$, $\sigma^\circ(l\star(2)) = S_2'$, and $\theta_{l\star(1)} = \theta_{l\star(2)} = \theta_l$. Then (b1) and (b2) trivially hold. (b3) is checked in a similar way to the case of $\mathtt{ap}_{\neg v}$.

The case $\phi(l) = (S_1, \Gamma_l \vdash f(y) : C)$ that is an instance of $\mathtt{ap}_v$, where $S_1 = \Gamma_l \vdash f : B \to C$, $y : B \in \Gamma_l$ and $y \neq \boldsymbol{x}$. We can define $\phi^\circ(l) = (S_1', \Gamma_l^\circ \vdash f[\theta_l](\theta_l(y)) : C)$ as $\mathtt{ap}_v$, where $S_1' = \Gamma_l^\circ \vdash f[\theta_l] : B \to C$, since $\theta_l(y) : B \in (\Gamma_l)_{\overline{\boldsymbol{x}}}[\theta_l] \subsetneq \Delta_l \sharp (\Gamma_l)_{\overline{\boldsymbol{x}}}[\theta_l] \sim \Gamma_l^\circ$ holds by (b1). Also define $\sigma^\circ(l\star(1)) = S_1'$ and $\theta_{l\star(1)} = \theta_l$. Then (b1) and (b2) trivially hold. (b3) is checked in a similar way to the case $\mathtt{ap}_{\neg v}$.

The case $\phi(l) = (\Gamma_l, z : C \vdash b : B, \Gamma_l \vdash \lambda z.b : C \to B)$ that is an instance of λ'. By the assumption, we have θ_l and $\sigma^\circ(l) = \Gamma^\circ \vdash (\lambda z.b)[\theta_l] : C \to B$ that satisfy (b1). Let $\theta_l = \{\vec{u}/\vec{x}\}$. We consider two subcases.

- The first subcase is when $z \notin \mathrm{FV}(\vec{u})$. We have $(\lambda z.b)[\theta_l] = \lambda z.(b[(\theta_l)_{\overline{z}}])$. Then define $\phi^\circ(l) = (\Gamma_l^\circ, z : C \vdash b[(\theta_l)_{\overline{z}}] : B, \Gamma_l^\circ \vdash \lambda z.(b[(\theta_l)_{\overline{z}}]) : C \to B)$ as λ. Note that $\Gamma_l^\circ, z : C$ is a context because $z : C \notin \Delta_l \sharp (\Gamma_l)_{\overline{\boldsymbol{x}}}[\theta_l] \sim \Gamma_l^\circ$ holds by (b1), $z \notin \mathrm{FV}(\Gamma_l)$ and $z \notin \mathrm{FV}(\vec{u}) \supseteq \mathrm{FV}(\Delta_l)$ (recall that Δ_l is Δ when $\boldsymbol{x} : A \in \Gamma_l$, and is $\emptyset$ otherwise). Define $\sigma^\circ(l\star(1)) = \Gamma_l^\circ, z : C \vdash b[(\theta_l)_{\overline{z}}] : B$ and $\theta_{l\star(1)} = (\theta_l)_{\overline{z}}$. We have (b2) by the definition. We also have (b1) by $(\Gamma_l^\circ, z : C) \sim (\Delta_l \sharp (\Gamma_l)_{\overline{\boldsymbol{x}}}[\theta_l], z : C) = \Delta_l \sharp (\Gamma_l, z : C)_{\overline{\boldsymbol{x}}}[\theta_{l\star(1)}]$. To check (b3), assume that $\tilde{k}, b$ is a successor of $k, \lambda z.b$ and k is not a named index for $\boldsymbol{x}$. Then (i) both k and $\tilde{k}$ are named indexes for some variable $y^D \in \mathrm{Dom}(\Gamma_l)$, (ii) both k and $\tilde{k}$ are unnamed indexes (not for C), or (iii) k is an unnamed indexes for C and $\tilde{k}$ is a named index for z. The cases (i) and (ii) can be checked in a similar way to the case of $\mathtt{ap}_{\neg v}$. The case (iii) is checked by taking k' and $\tilde{k}'$ as the same indexes as k and $\tilde{k}$, respectively.
- The second subcase is when $z \in \mathrm{FV}(\vec{u})$. We have $(\lambda z.b)[\theta_l] = \lambda z'.(b[(\theta_l)_{\overline{z}}, z'/z])$, where $z' \notin \mathrm{FV}(b, \vec{u})$. Then define $\phi^\circ(l) = (\Gamma_l^\circ, z' : C \vdash b[(\theta_l)_{\overline{z}}, z'/z] : B, \Gamma_l^\circ \vdash \lambda z'.(b[(\theta_l)_{\overline{z}}, z'/z]) : C \to B)$ as λ. We need to check that $\Gamma_l^\circ, z' : C$ is a context. It is shown by $z' : C \notin \Delta_l \sharp (\Gamma_l)_{\overline{\boldsymbol{x}}}[\theta_l] \sim \Gamma_l^\circ$ using (b1), $z' \notin \mathrm{FV}(\lambda z.b) = \mathrm{FV}(\Gamma_l)$ and $z' \notin \mathrm{FV}(\vec{u}) \supseteq \mathrm{FV}(\Delta_l)$. Define $\sigma^\circ(l\star(1)) = \Gamma_l^\circ, z' : C \vdash b[(\theta_l)_{\overline{z}}, z'/z] : B$ and $\theta_{l\star(1)} = (\theta_l)_{\overline{z}} \cup \{z'/z\}$. We have (b2) by the definition. We also have (b1) by $(\Gamma_l^\circ, z' : C) \sim (\Delta_l \sharp (\Gamma_l)_{\overline{\boldsymbol{x}}}[\theta_l], z' : C) = \Delta_l \sharp (\Gamma_l, z : C)_{\overline{\boldsymbol{x}}}[\theta_{l\star(1)}]$. Checking (b3) is done in a similar way to that of the first subcase.

The case $\phi(l) = (S_1, S_2, \Gamma_l \vdash \mathtt{cond}(f, g) : N \to C)$, where $S_1 = \Gamma_l \vdash f : C$, $S_2 = \Gamma_l \vdash g : N \to C$, as an instance of $\mathtt{cond}$. Define $\phi^\circ(l) = (S_1', S_2', \Gamma_l^\circ \vdash \mathtt{cond}(f[\theta_l], g[\theta_l]) : N \to C)$ as $\mathtt{cond}$, where $S_1' = \Gamma_l^\circ \vdash f[\theta_l] : C$ and $S_2' = \Gamma_l^\circ \vdash g[\theta_l] : N \to C$. Also define $\sigma^\circ(l\star(1)) = S_1'$, $\sigma^\circ(l\star(2)) = S_2'$, and $\theta_{l\star(1)} = \theta_{l\star(2)} = \theta_l$. Then (b1) and (b2) trivially hold. (b3) is checked in a similar way to the case of $\mathtt{ap}_{\neg v}$.

The case $\phi(l) = (\Gamma_l \vdash 0 : N)$, as an instance of 0. Define $\phi^\circ(l) = (\Gamma_l^\circ \vdash 0 : N)$ as 0. Since $l\star(i) \notin T_t$, (b1), (b2), and (b3) trivially hold.

The case $\phi(l) = (\Gamma_l \vdash t : N, \Gamma_l \vdash \mathtt{S}(t) : N)$, as an instance of $\mathtt{S}$. Define $\phi^\circ(l) = (\Gamma_l^\circ \vdash t[\theta_l] : N, \Gamma_l^\circ \vdash \mathtt{S}(t[\theta_l]) : N)$ as $\mathtt{S}$. Also define $\sigma^\circ(l\star(1)) = \Gamma_l^\circ \vdash t : N$ and $\theta_{l\star(1)} = \theta_l$. Then (b1) and (b2) trivially hold. (b3) is checked in a similar way to the case of $\mathtt{ap}_{\neg v}$.

Therefore our construction of Π° is completed.

Lemma 6.3 *1. If $\Pi : \Gamma \vdash f(a) : A$ and Π satisfies the global trace condition, then there exist Π_1, Π_2 and B such that $\Pi_1 : \Gamma \vdash f : B \to A$, $\Pi_2 : \Gamma \vdash a : B$, and both Π_1 and Π_2 satisfy the global trace condition.*

2. If $\Pi : \Gamma \vdash \lambda x^T.b : A$, where $x \notin \mathrm{FV}(\Gamma)$, and Π satisfies the global trace condition, then there exist Π_1 and B such that $\Pi_1 : \Gamma, x : T \vdash b : B$ and $A = T \to B$, and Π_1 satisfies the global trace condition.

3. If $\Pi : \Gamma \vdash \mathtt{cond}(f, g) : A$ and Π satisfies the global trace condition, then there exist Π_1, Π_2 and B such that $\Pi_1 : \Gamma \vdash f : B$, $\Pi_2 : \Gamma \vdash g : N \to B$, $A = N \to B$, and both Π_1 and Π_2 satisfy the global trace condition.

4. If $\Pi : \Gamma \vdash \mathtt{S}(t) : A$ and $\mathtt{S}(t) \in \mathtt{GTC}$, then there exists Π_1 such that $\Pi_1 : \Gamma \vdash t : N$, $A = N$, and Π_1 satisfies the global trace condition.

Proof We first claim that if $t \in \mathtt{GTC}$ and $\Pi : \Gamma \vdash t : A$ with $\Pi = (T, \phi)$, then there exists $l \in T$ such that $\phi(l) \neq \mathtt{weak}$ and $\phi(m) = \mathtt{weak}$ for all $m < l$. Because, if not, the only infinite path in Π is the consective use of $\mathtt{weak}$, which does not contain progressing trace, this contradicts with $t \in \mathtt{GTC}$.

We show the point 1.. Assume that $\Pi : \Gamma \vdash f(a) : A$ with $\Pi = (T, \phi)$ and $f(a) \in \mathtt{GTC}$. By the claim, take $l \in T$ such that $\phi(l) \neq \mathtt{weak}$ and $\phi(m) = \mathtt{weak}$ for all $m < l$. Let $\Gamma' \vdash f(a) : A$ be the conclusion of $\phi(l)$. Then $\Gamma' \subsetneq \Gamma$ holds by the transitivity of $\subsetneq$. Now $\phi(l)$ is $\mathtt{ap}_v$ or $\mathtt{ap}_{\neg v}$. In the former case $\phi(l) = \mathtt{ap}_v$, we have $\Pi\lceil l\star(1) : \Gamma' \vdash f : B \to A$ for some B, and $a = x^B \in \Gamma'$. Hence we have a proof $\Pi_1 : \Gamma \vdash f : B \to A$ by $\mathtt{weak}$. We also obtain $\Pi_2 : \Gamma \vdash a : B$ by $a = x^B \in \Gamma'$ and $\mathtt{weak}$. In the latter case $\phi(l) = \mathtt{ap}_{\neg v}$, we have $\Pi\lceil l\star(1) : \Gamma' \vdash f : B \to A$ and $\Pi\lceil l\star(2) : \Gamma' \vdash a : B$ for some B. Hence we have a

proof $\Pi_1 : \Gamma \vdash f : B \to A$ and $\Pi_2 : \Gamma \vdash a : B$ by $\mathtt{weak}$. In both cases, if $t \in \mathtt{GTC}$, we have $f \in \mathtt{GTC}$ and $a \in \mathtt{GTC}$ by the construction of Π_1 and Π_2.

The points 2., 3., and 4. are shown similarly.

Theorem 6.4 (Subject reduction) *Assume that $\Pi_t : \Gamma \vdash t : A$, Π_t satisfies the global trace condition, and $t \mapsto u$. Then there exists Π_u such that $\Pi_u : \Gamma \vdash u : A$ and Π_u satisfies the global trace condition.*

Proof By the definition of $t \mapsto u$, there is a context $\hat{t}[-]$ such that $t = \hat{t}[t_0]$, $u = \hat{t}[u_0]$, and $t_0 \mapsto_\Box u_0$, where $\Box \in \{\beta, \mathtt{cond}\}$. Since t_0 is a subterm of t, there is l such that $\Pi_t\lceil l : \Gamma_0 \vdash t_0 : A_0$. Note that $\Pi_t\lceil l$ satisfies the global trace condition, since it is a subtree of Π_t, which satisfies the global trace condition. Then, if we have $\Pi'_u : \Gamma_0 \vdash u_0 : A_0$ that satisfies the global trace condition, the tree obtained from Π_t by replacing the subtree $\Pi_t\lceil l$ by Π'_u is also a proof of $\Gamma \vdash \hat{t}[u_0] : A$ that satisfies the global trace condition. Hence it is enough to show the following:

(a) If $\Pi : \Gamma \vdash (\lambda x^A.b)(a) : B$ and it satisfies the global trace condition, then $\Pi' : \Gamma \vdash b[a/x] : B$ for some Π' that satisfies the global trace condition.

(b) If $\Pi : \Gamma \vdash \mathtt{cond}(f, g)(0) : B$ and it satisfies the global trace condition, then $\Pi' : \Gamma \vdash f : B$ for some Π' that satisfies the global trace condition.

(c) If $\Pi : \Gamma \vdash \mathtt{cond}(f, g)(\mathtt{S}(t)) : B$ and it satisfies the global trace condition, then $\Pi' : \Gamma \vdash g(t) : B$ for some Π' that satisfies the global trace condition.

(b) is shown immediately by Lemma 6.3 3.. We show (c). Assume $\Pi : \Gamma \vdash \mathtt{cond}(f, g)(\mathtt{S}(t)) : B$. Then by 1., 3., and 4. of Lemma 6.3, we have $\Pi_1 : \Gamma \vdash g : N \to B$ and $\Pi_2 : \Gamma \vdash t : N$, where Π_1 and Π_2 satisfy the global trace condition. Hence, by applying $\mathtt{ap}_{\neg v}$ or $\mathtt{ap}_v$ to Π_1 and Π_2, we have a proof $\Pi' : \Gamma \vdash g(t) : B$ that satisfies the global trace condition. In order to show (a), assume $\Pi : \Gamma \vdash (\lambda x^A.b)(a) : B$. Then by Lemma 6.3 1., we have $\Pi_1 : \Gamma \vdash \lambda x^A.b : A \to B$ and $\Pi_2 : \Gamma \vdash a : A$, where Π_1 and Π_2 satisty the global trace condition. Then, by Lemma 6.1, we have $\Pi'_1 : \Gamma_{\mathrm{FV}(\lambda x.b)} \vdash \lambda x^A.b : A \to B$, where Π' satisfies the global trace condition. By Lemma 6.3 2., we have a proof $\Pi''_1 : \Gamma_{\mathrm{FV}(\lambda x.b)}, x : A \vdash b : B$ that satisfies the global trace condition. Hence, by the substitution lemma, we have a proof $\Pi' : \Gamma \vdash b[a/x] : B$ that satisfies the global trace condition, as we wished.

Using the subject reduction theorem, variable renaming is shown to be admissible in our system.

Proposition 6.5 *1. Let θ be a renaming. If $\Pi : \Gamma \vdash t : A$ and $\Gamma[\theta]$ is a context, then $\Pi' : \Gamma[\theta] \vdash t[\theta] : A$ for some Π'. Moreover, if Π satisfies the global trace condition, so is Π'.*

2. If $\Pi : \Gamma \vdash t : A$ and t' is an α-equivalent term of t, then $\Pi' : \Gamma \vdash t' : A$ for some Π'. Moreover, if Π satisfies the global trace condition, so is Π'.

Proof We show the point 1.. It is enough to show the claim for a single renaming $\theta = \{y'/y\}$. Then, by the assumption, we have a proof Π_1 of $\Gamma[y'/y] \vdash (\lambda y^B.t)y' : A$ such that Π_1 satisfies the global trace condition if Π satisfies it. Hence we have $\Pi' : \Gamma[y'/y] \vdash t[y'/y] : A$ by the subject reduction theorem such that Π' satisfies the global trace condition if Π satisfies it.

Next we show the point 2.. It is enough to show when $t = \lambda x.b$ and $t' = \lambda x'.(b[x'/x])$, where $x' \notin \mathrm{FV}(\lambda x.b)$. Let Π be a proof of $\Gamma \vdash \lambda x.b : A \to B$. Then we have $\Pi_1 : \Gamma_{\mathrm{FV}(\lambda x.b)} \vdash \lambda x.b : A \to B$ by Lemma 6.1, and also have $\Pi_2 : \Gamma_{\mathrm{FV}(\lambda x.b)}, x' : B \vdash b[x'/x] : B$ by the subject reduction theorem. Hence we have $\Pi' : \Gamma \vdash \lambda x'.b[x'/x] : A \to B$ by the rules λ and $\mathtt{weak}$. Note that Π' satisfies the global trace condition if Π satisfies it.

7 Weak Church-Rosser for safe reductions

In this section we prove a result for the n-safe part of a term, which we call "*unicity of the n-safe part of the safe normal form up to $\approx_{n,\mathtt{cond}}$*". By this we mean: for all $t, u, v \in \mathtt{GTC}$, if $t \mapsto_{n,\mathtt{cond},\mathtt{safe}} u$ and $t \mapsto_{n,\mathtt{cond},\mathtt{safe}} v$ and u, v are n-safe-normal then $u \approx_{n,\mathtt{cond}} v$ holds.

Our first idea (wrong) is to prove a full Church-Rosser property for $\mathbb{\Lambda}$: for all $t, u, v \in \mathbb{\Lambda}$, if $t \mapsto u$ and $t \mapsto v$ then for some $w \in \mathbb{\Lambda}$ we have $u \mapsto w$ and $v \mapsto w$. This property is false: for some $t \in \mathbb{\Lambda}$, finding a common reduction of u, v takes infinitely many steps. This even in the case $t \in \mathtt{CT}\text{-}\lambda$, as the next example shows.

Example 7.1 (Failure of Church-Rosser for $\mathtt{CT}\text{-}\lambda$) *Let $b = \mathtt{cond}(x^N, b) : N \to N$ a normal form and $t = (\lambda x^N.b)(r) : N \to N$, where $r = (\lambda x^N.x^N)(3)$ is some redex.*

We have $b[r/x](n) \mapsto r \mapsto 3$ for all numerals n, therefore t and $\lambda\_.3$ are extensionally equal, however $t \not\mapsto \lambda\_.3$. We have $t \in \mathtt{CT}\text{-}\lambda$. Indeed,

1. *t is regular by construction.*
2. *We have $t \in \mathtt{GTC}$, because the unique infinite path of t is $t, \lambda x^N.b, b, b, b, \ldots$, and the unique unnamed argument of $b : N \to N$ in the path progresses infinitely many times.*

Now consider the reductions: $t \mapsto b[r/x^N]$ and $t \mapsto (\lambda x^N.b)(3)$. We expect $b[3/x^N]$ as common normal form. But we have $b[r/x^N] = \mathtt{cond}(r, b[r/x^N]$, that is, we have replicated the redex r infinitely many times in $b[r/x^N]$. Therefore to reduce $b[r/x^N]$ to $b[3/x^N]$ takes infinitely many steps, and for no finite reduction we have $b[r/x^N] \mapsto b[3/x^N]$.

We proved that Church-Rosser is false for $\mathtt{CT}\text{-}\lambda$.

In the following we will work in the n-safe level of $t \in \mathtt{GTC}$. The global trace condition guarentees the finiteness of the n-safe level for each n.

Lemma 7.1 *For any $n \geq 0$, the set of the n-safe level of $t \in \mathtt{GTC}$ is finite.*

Proof Let $t \in \mathtt{GTC}$. We show the finiteness of any n-level of t by proof of contradiction. Assume that the finiteness fails. Then take the least number n such that the n-safe level of t is infinite.

The case of $n = 0$. Since the binary tree obtained from t restricting to the 0-safe level of t contains infinite nodes, we can take an infinite path $t, t_1, t_2, \ldots$ of the tree by using König's lemma. Then this infinite path does not contain a progressing trace since any t_i cannot be the right subterm of $\mathtt{cond}$. This contradicts $t \in \mathtt{GTC}$.

The case of $n > 0$. By the leastness of n, for any $n' < n$, all n'-level of t are finite. We can also take an infinite path from the binary tree that is a restriction of t with nodes until the n-safe level of t. The path has the form $\vec{t_0}, \vec{t_1}, \ldots, \vec{t_n}$, where each $\vec{t_i}$ is a sequence of nodes of the i-safe level, and only $\vec{t_n}$ is infinite. Then this infinite path does not contain a progressing trace as in the case of $n = 0$. This also contradicts $t \in \mathtt{GTC}$.

Hence we have the finiteness of the n-safe level of $t \in \mathtt{GTC}$, as we wished.

We write $||t||_n$ for the number of the n-safe level of $t \in \mathtt{GTC}$.

We define a n-safe level context $\mathtt{C}_n$ with holes (written $\cdot$) that will be filled with terms in $\mathbb{\Lambda}$, and whose positions are at the n-safe level.

Definition 7.2 (Safe level context) *We define $\mathtt{C}_{-1} = \cdot$. For $n \geq 0$, the n-safe level context, written $\mathtt{C}_n$, is inductively defined as follows:*

$$\mathtt{C}_n ::= x^T \mid 0 \mid \mathtt{C}_n\mathtt{C}_n \mid \lambda x^T.\mathtt{C}_n \mid \mathtt{S}(\mathtt{C}_n) \mid \mathtt{cond}(\mathtt{C}_n, \mathtt{C}_{n-1}).$$

Multiple holes may appear in a context and each holes are distinguished. The resulting term obtained by filling k-holes in $\mathtt{C}_n$ with terms $t_1, \ldots, t_k$ is written $\mathtt{C}_n[t_1, \ldots, t_k]$. Note that filling holes with terms is not substitution, but just putting terms at the positions of holes, namely, for example, the result of filling the unique hole in $\lambda x.\mathtt{cond}(0, \cdot)$ with x is $\lambda x.\mathtt{cond}(0, x)$.

The finiteness of the n-safe level of $t \in \mathtt{GTC}$ enables to split t by a n-safe level context and terms that apper at level $> n$ as stated in the next lemma.

Lemma 7.3 *1. For any $\mathtt{C}_n$, there uniquely exists $(\mathtt{C}_0, \mathtt{C}^1_{n-1}, \ldots, \mathtt{C}^k_{n-1})$ such that $\mathtt{C}_0$ has k-holes and $\mathtt{C}_n = \mathtt{C}_0[\mathtt{C}^1_{n-1}, \ldots, \mathtt{C}^k_{n-1}]$, namely $\mathtt{C}_n[\vec{t_1}, \ldots, \vec{t_k}] = \mathtt{C}_0[\mathtt{C}^1_{n-1}[\vec{t_1}], \ldots, \mathtt{C}^k_{n-1}[\vec{t_k}]]$ holds for any $\vec{t_1}, \ldots, \vec{t_k}$.*

2. Let $t \in \mathtt{GTC}$ and $n \geq 0$. There uniquely exists $(\mathtt{C}_n, t_1, \ldots, t_k)$ such that $\mathtt{C}_n$ has k-holes, $t = \mathtt{C}_n[t_1, \ldots, t_k]$, and all $t_1, \ldots, t_k$ appear at the $(n+1)$-safe level of t.

Proof The point 1. is shown by induction on the construction of $\mathtt{C}_n$. If $\mathtt{C}_n$ is x^T or 0, then define $\mathtt{C}_0$ by the same one as $\mathtt{C}_n$. For the case of $\mathtt{C}_n = \lambda x^T.\mathtt{C}'_n$, we have a unique $(\mathtt{C}'_0, \overrightarrow{\mathtt{C}'_{n-1}})$ such that $\mathtt{C}'_n = \mathtt{C}'_0[\overrightarrow{\mathtt{C}'_{n-1}}]$. Then $(\lambda x^T.\mathtt{C}'_0, \overrightarrow{\mathtt{C}'_{n-1}})$ satisfies the expected condition for $\mathtt{C}_n$ of this case. The cases of $\mathtt{C}_n = \mathtt{C}'_n\mathtt{C}''_n$ and $\mathtt{C}_n = \mathtt{S}(\mathtt{C}'_n)$ are shown similarly by using the induction hypothesis. We show the case $\mathtt{C}_n = \mathtt{cond}(\mathtt{C}'_n, \mathtt{C}''_{n-1})$. By the induction hypothesis, there exists a unique $(\mathtt{C}'_0, \overrightarrow{\mathtt{C}'_{n-1}})$ such that $\mathtt{C}'_n = \mathtt{C}'_0[\overrightarrow{\mathtt{C}'_{n-1}}]$. Then $(\mathtt{cond}(\mathtt{C}'_0, \cdot), \overrightarrow{\mathtt{C}'_{n-1}}, \mathtt{C}''_{n-1})$ satisfies the expected condition for $\mathtt{C}_n$ of this case.

The point 2. is shown by induction on n using 1..

We first show the case of $n = 0$ by induction on the number of the 0-safe level of t. If $t = x^T$ or $t = 0$, it is shown by taking $\mathtt{C}_0$ as t. If $t = \lambda x^T.t'$, by the induction hypothesis, there uniquely exists $(\mathtt{C}'_0, \vec{t'})$ that satisfies the condition for t'. Then $(\lambda x^T.\mathtt{C}'_0, \vec{t'})$ satisfies the expected condition for $\lambda x^T.t'$. If $t = t't''$ or $t = \mathtt{S}(t')$, it is also shown by the induction hypothesis. If $t = \mathtt{cond}(t', f)$, by the induction hypothesis, there uniquely exists $(\mathtt{C}'_0, \vec{t'})$ that satisfies the condition for t'. Then $(\mathtt{cond}(\mathtt{C}'_0, \cdot), \vec{t'}, f)$ satisfies the expected condition for $\mathtt{cond}(t', f)$.

Then we show the case of $n > 0$. By the result of the case of $n = 0$, there uniquely exists $(\mathtt{C}_0, t_1, \ldots, t_k)$ that satisfies $t = \mathtt{C}_0[t_1, \ldots, t_k]$ and each t_i appears at the 1-safe level of t. For each i, by applying the induction hypothesis to t_i, we have unique $(\mathtt{C}^i_0, \vec{t'_i})$ satisfies $t_i = \mathtt{C}^i_0[\vec{t'_i}]$ and $\vec{t'_i}$ appear at the $(n-1)$-safe level of t_i. Then $(\mathtt{C}_0[\mathtt{C}^1_{n-1}, \ldots, \mathtt{C}^k_{n-1}], \vec{t'_1}, \ldots, \vec{t'_k})$ satisfies the expected condition for t. Its uniqueness is obtained by using the uniqueness of $\mathtt{C}_0$ and $\mathtt{C}^i_{n-1}$, and the point 1..

We define the notion "n-safe equality" that intuitively means two terms are approximately equal excepting for deeper safe levels more than n.

Definition 7.4 (n-safe equality) *Let $\Gamma \vdash t_1 : A$ and $\Gamma \vdash t_2 : A$. The terms t_1 and t_2 are "n-safe equal" (written $\Gamma \vdash t_1 \approx_{n,\mathtt{cond}} t_2 : A$) is defined by after possibly renaming the bound variables of t_1 and t_2, they have the forms $\mathtt{C}_n[\vec{u}_1]$ and $\mathtt{C}_n[\vec{u}_2]$ with some terms $\vec{u}_1$ and $\vec{u}_2$. We sometimes write $t_1 \approx_{n,\mathtt{cond}} t_2$ instead of $\Gamma \vdash t_1 \approx_{n,\mathtt{cond}} t_2 : A$ when Γ and A are clear from the context.*

The n-safe equality is necessary to fill "gaps" after n-safe reductions from the same term. For example, let t be $(\lambda x^{N\to N}.\mathtt{cond}(0, x))(IS)$, where $I = \lambda z^{N\to N}.z$ and $S = \lambda n^N.\mathtt{S}(n)$. For the two 0-safe reducuctions $t \mapsto^*_{0,\mathtt{safe}} \mathtt{cond}(0, S)$ and $t \mapsto^*_{0,\mathtt{safe}} \mathtt{cond}(0, IS)$, we have $\mathtt{cond}(0, IS) \approx_0 \mathtt{cond}(0, S)$ since $\mathtt{cond}(0, IS) \mapsto_{0,\mathtt{safe}} \mathtt{cond}(0, S)$ does not hold.

Note that $\approx_{n,\mathtt{cond}}$ is an equivalence relation. Moreover it is closed under substitution.

Lemma 7.5 *If $\Gamma \vdash u_1 \approx_{n,\mathtt{cond}} u_2 : A$ and $\Gamma, x^A : A \vdash t_1 \approx_{n,\mathtt{cond}} t_2 : B$, then $\Gamma \vdash t_1[u_1/x^A] \approx_{n,\mathtt{cond}} t_2[u_2/x^A] : B$.*

Proof By the substitution lemma, we have $\Gamma \vdash t_i[u_i/x^A] : B$ for $i \in \{1, 2\}$. Thus we need to show the equality part. It is easily checked that $t_1[u_1/x^A] \approx_{-1} t_2[u_2/x^A]$ by the definition of the safe equality. For $n \geq 0$, by $t_1 \approx_{n,\mathtt{cond}} t_2$, there exists $(\mathtt{C}_n, \vec{u_1}, \vec{u_2})$ such that $t_1 = \mathtt{C}_n[\vec{v_1}]$ and $t_2 = \mathtt{C}_n[\vec{v_2}]$. We show $\mathtt{C}_n[\vec{v_1}][u_1/x^A] \approx_{n,\mathtt{cond}} \mathtt{C}_n[\vec{v_2}][u_2/x^A]$ by induction on $\mathtt{C}_n$.

The case of $\mathtt{C}_n = x^A$ is shown by $x^A[u_1/x^A] = u_1 \approx_{n,\mathtt{cond}} u_2 = x^A[u_2/x^A]$. The cases of $\mathtt{C}_n = 0$ and $\mathtt{C}_n = z^C \neq x^A$ are shown by $t_1[u_1/x^A] = t_1 \approx_{n,\mathtt{cond}} t_2 = t_2[u_2/x^A]$. The case of $\mathtt{C}_n = \lambda z^C.\mathtt{C}'_n$ with $z^C \notin \mathrm{FV}(u_1, u_2)$ by renaming is shown by $(\lambda z^C.\mathtt{C}'_n[\vec{v_1}])[u_1/x^A] = \lambda z^C.(\mathtt{C}'_n[\vec{v_1}][u_1/x^A]) \approx_{n,\mathtt{cond}} \lambda z^C.(\mathtt{C}'_n[\vec{v_2}][u_2/x^A]) = (\lambda z^C.\mathtt{C}'_n[\vec{v_2}])[u_2/x^A]$ by the induction hypothesis. The cases of $\mathtt{C}_n = \mathtt{C}'_n\mathtt{C}''_n$ and $\mathtt{C}_n = \mathtt{S}(\mathtt{C}'_n)$ are also shown by the induction hypothesis. We show the case of $\mathtt{C}_n = \mathtt{cond}(\mathtt{C}'_n, \mathtt{C}'_{n-1})$. If $n = 0$, by induction hypothesis, we have $\mathtt{C}_0[\vec{v_1}][u_1/x^A] = \mathtt{cond}(\mathtt{C}'_0[\vec{v_1}][u_1/x^A], \mathtt{C}'_{-1}[\vec{v_1}][u_1/x^A]) \approx_0 \mathtt{cond}(\mathtt{C}'_0[\vec{v_2}][u_2/x^A], \mathtt{C}'_{-1}[\vec{v_2}][u_2/x^A]) = \mathtt{C}_0[\vec{v_2}][u_2/x^A]$ since we do not care the right-hand side of $\mathtt{cond}$ at the 0-safe level. If $n > 0$, by induction hypothesis, we have $\mathtt{C}_n[\vec{v_1}][u_1/x^A] = \mathtt{cond}(\mathtt{C}'_n[\vec{v_1}][u_1/x^A], \mathtt{C}'_{n-1}[\vec{v_1}][u_1/x^A]) \approx_{n,\mathtt{cond}} \mathtt{cond}(\mathtt{C}'_n[\vec{v_2}][u_2/x^A], \mathtt{C}'_{n-1}[\vec{v_2}][u_2/x^A]) = \mathtt{C}_n[\vec{v_2}][u_2/x^A]$. Hence we obtain the result as we wished.

The main theorem of this section is a weak form of Church-Rosser of the n-safe reductions that holds up to the n-equalities.

Theorem 7.6 (Weak Church-Rosser of n-safe reduction modulo n-safe equality) *Let $t \in \mathtt{GTC}$. If $t_1 \leftarrow\!\!\mapsto^*_{n,\mathtt{safe}} t \mapsto^*_{n,\mathtt{safe}} t_2$, then there exist t'_1 and t'_2 such that $t_1 \mapsto^*_{n,\mathtt{safe}} t'_1 \approx_{n,\mathtt{cond}} t'_2 \leftarrow\!\!\mapsto^*_{n,\mathtt{safe}} t_2$.*

This theorem will be proved by a variant of the parallel reduction technique.

Definition 7.7 (Safe parallel reduction) *Let $n \geq -1$. We define the n-safe parallel reduction relation $\Rightarrow_n$ on the $\mathtt{GTC}$ terms. The $\Rightarrow_{-1}$ is defined by $t \Rightarrow_{-1} t$ for all $t \in \mathtt{GTC}$. For $n \geq 0$, the relation $\Rightarrow_n$ is inductively defined as follows:*

(id) $t \Rightarrow_n t$.

($\mathtt{S}$) *If* $t \Rightarrow_n t'$, *then* $\mathtt{S}(t) \Rightarrow_n \mathtt{S}(t')$.

(λ) *If* $t \Rightarrow_n t'$, *then* $\lambda x^T.t \Rightarrow_n \lambda x^T.t'$.

(ap) *If* $f \Rightarrow_n f'$ *and* $a \Rightarrow_n a'$, *then* $f(a) \Rightarrow_n f'(a')$.

($\mathtt{cond}$) *If* $a \Rightarrow_n a'$ *and* $f \Rightarrow_{n-1} f'$, *then* $\mathtt{cond}(a,f) \Rightarrow_n \mathtt{cond}(a',f')$.

(β) *If* $t \Rightarrow_n t'$ *and* $u \Rightarrow_n u'$, *then* $(\lambda x^T.t)u \Rightarrow_n t'[u'/x]$.

($\mathtt{cond}\,0$) *If* $a \Rightarrow_n a'$, *then* $\mathtt{cond}(a,f)(0) \Rightarrow_n a'$.

($\mathtt{cond}\,\mathtt{S}$) *If* $f \Rightarrow_{n-1} f'$ *and* $u \Rightarrow_n u'$, *then* $\mathtt{cond}(a,f)(\mathtt{S}(u)) \Rightarrow_n f'(u')$.

The relation $\Rightarrow_{-1}$ is a special case to simplify the rules ($\mathtt{cond}$) and ($\mathtt{cond}\,\mathtt{S}$). For $n \geq 0$, if t is reduced to u by the n-safe parallel reduction, redexes that appear in the n-safe level of t can be reduced. For example, $\mathtt{cond}(I0, \mathtt{cond}(I0, IS)) \Rightarrow_n \mathtt{cond}(0, \mathtt{cond}(0, S))$ holds if $n = 2$, but does not hold if $n = 1$. The case (id) cannot be replaced by $x^T \Rightarrow_n x^T$ and $0 \Rightarrow_n 0$ expecting (id) can be derived from them, but it is not the case in our setting because we are working on the infinite terms.

The relation $\Rightarrow_n$ satisfies the following basic properties.

Lemma 7.8 *Let $t, u \in \mathtt{GTC}$. The following properties hold.*

1. *$n < n'$ and $t \Rightarrow_n t'$ implies $t \Rightarrow_{n'} t'$ (monotonicity).*
2. *If $t \mapsto_{n,\mathtt{cond},\mathtt{safe}} t'$, then $t \Rightarrow_n t'$.*
3. *If $t \Rightarrow_n t'$, then $t \mapsto^*_{n,\mathtt{safe}} t'$.*
4. *If $t \Rightarrow_n t'$ and $u \Rightarrow_n u'$, then $t[u/x^A] \Rightarrow_n t'[u'/x^A]$.*

Proof The point 1. is shown by induction on $\Rightarrow_n$.

We show the point 2.. For t and t' such as $t \mapsto_{n,\mathtt{cond},\mathtt{safe}} t'$ that reduces a redex t_R occuring in t, define $r(t,t')$ as the length of the path from t to t_R. The proof is done by induction of $r(t,t')$. The case of $r(t,t') = 0$, namely $t = t_R$, we have $t \Rightarrow_n t'$ by the rule (β), ($\mathtt{cond}\,0$), or ($\mathtt{cond}\,\mathtt{S}$). The case of $r(t,t') > 0$ is shown by the case analysis of the form of t. The case of $t = \mathtt{S}(t_1)$ and t_R appears in t_1. The reduction is $\mathtt{S}(t_1) \mapsto_{n,\mathtt{cond},\mathtt{safe}} \mathtt{S}(t'_1) = t'$ with $t_1 \mapsto_{n,\mathtt{cond},\mathtt{safe}} t'_1$. Then we have $t_1 \Rightarrow_n t'_1$ by the induction hypothesis with $r(t_1,t'_1) < r(t,t')$. Hence $\mathtt{S}(t_1) \Rightarrow_n \mathtt{S}(t'_1)$ by ($\mathtt{S}$). The cases of $t = t_1 t_2$ and $t = \lambda x^A.t_1$ are also shown by the induction hypothesis. The case of $t = \mathtt{cond}(a,f)$ and t_R appears in a is shown by the induction hypothesis. The case of $t = \mathtt{cond}(a,f)$ and t_R appears in f. The reduction is $\mathtt{cond}(a,f) \mapsto_{n,\mathtt{cond},\mathtt{safe}} \mathtt{cond}(a,f') = t'$ with $f \mapsto_{n-1,\mathtt{safe}} f'$. Then we have $f \Rightarrow_{n-1} f'$ by the induction hypothesis with $r(f,f') < r(t,t')$. Hence $\mathtt{cond}(a,f) \Rightarrow_n \mathtt{cond}(a,f')$ by ($\mathtt{cond}$).

The point 3. is shown by induction on $\Rightarrow_n$.

To prove the point 4., we first show that $u \Rightarrow_n u'$ implies $t[u/x^A] \Rightarrow_n t[u'/x^A]$ by induction on $||t||_n$. The case $t = x^A$ is shown by the assumption $u \Rightarrow_n u'$. The cases $t = y^B \neq x^A$ and $t = 0$ are shown by (id). The case $t = y^B \neq x^A$ is shown by (id). The case $t = \lambda y^B.t'$ with $y^B \notin \mathrm{FV}(u,u')$ by renaming is shown by $(\lambda y^B.t')[u/x^A] = \lambda y^B.(t'[u/x^A]) \Rightarrow_n \lambda y^B.(t'[u'/x^A]) = (\lambda y^B.t')[u'/x^A]$ by the induction hypothesis and (λ). The cases $t = t_1 t_2$ and $t = \mathtt{S}(t')$ are also shown by the induction hypothesis. We show the case $t = \mathtt{cond}(a,f)$ and $n > 0$. Since $||a||_{n-1}, ||f||_n < ||\mathtt{cond}(a,f)||_n$, we have $a[u/x^A] \Rightarrow_n a[u'/x^A]$ and $f[u/x^A] \Rightarrow_{n-1} f[u'/x^A]$ by the induction hypothesis. Hence $\mathtt{cond}(a,f)[u/x^A] \Rightarrow_n \mathtt{cond}(a,f)[u'/x^A]$ by ($\mathtt{cond}$). The case $t = \mathtt{cond}(a,f)$ and $n = 0$ is shown similarly.

Then we show the the point 4. by induction on $\Rightarrow_n$. The case of (id) is already shown. The cases of ($\mathtt{S}$), (λ), (ap), ($\mathtt{cond}$), ($\mathtt{cond}\,0$), and ($\mathtt{cond}\,\mathtt{S}$) are shown by the induction hypothesis. We show the case

(β) with $t = (\lambda y^B.t_1)t_2$, where $y^B \notin \mathrm{FV}(u, u')$ by renaming, $t' = t'_1[t'_2/y^B]$, $t_1 \Rightarrow_n t'_1$, and $t_2 \Rightarrow_n t'_2$. By the induction hytpothesis, $t_1[u/x^A] \Rightarrow_n t'_1[u'/x^A]$ and $t_2[u/x^A] \Rightarrow_n t'_2[u'/x^A]$ hold. Then, by the substitution lemma (Lemma 2.8), $((\lambda y^B.t_1)t_2)[u/x^A] = (\lambda y^B.(t_1[u/x^A]))t_2[u/x^A] \Rightarrow_n t_1[u/x^A][t_2[u/x^A]/y^B] = t_1[t_2/y^B][u/x^A]$ as we wished.

The next lemma says that the n-equality simulates

Lemma 7.9 *Let $t_1, t_2 \in \mathtt{GTC}$ and $n \geq 0$. If $t_1 \approx_{n,\mathtt{cond}} t_2 \Rightarrow_n t'_2$, then there exists $t'_1 \in \mathtt{GTC}$ such that $t_1 \Rightarrow_n t'_1 \approx_{n,\mathtt{cond}} t'_2$.*

Proof The proof is done be induction on $t_2 \Rightarrow_n t'_2$.

The case of (id), namely $t_2 = t'_2$, is shown by taking t_1 as t'_1.

The case of $(\mathtt{S})$, namely $t_2 = \mathtt{S}(u_2) \Rightarrow_n \mathtt{S}(u'_2) = t'_2$ that is obtained from $u_2 \Rightarrow_n u'_2$. There is u_1 such that $t_1 = \mathtt{S}(u_1)$ and $u_1 \approx_{n,\mathtt{cond}} u_2$ by $t_1 \approx_{n,\mathtt{cond}} \mathtt{S}(u_2)$ with $n \geq 0$. Then, by the induction hypothesis, there exists $u'_1 \in \mathtt{GTC}$ such that $u_1 \Rightarrow_n u'_1 \approx_{n,\mathtt{cond}} u'_2$. This case is shown by taking $\mathtt{S}(u'_1)$ as t'_1, since $\mathtt{S}(u_1) \Rightarrow_n \mathtt{S}(u'_1) \approx_{n,\mathtt{cond}} \mathtt{S}(u'_2)$.

The cases of (λ), (ap), and $(\mathtt{cond})$ are shown similarly by using the induction hypothesis.

The case of $(\mathtt{cond}\,0)$, namely $t_2 = \mathtt{cond}(a_2, f_2)(0) \Rightarrow_n a'_2$ that is obtained from $a_2 \Rightarrow_n a'_2$. There are a_1, f_1 such that $t_1 = \mathtt{cond}(a_1, f_1)(0)$ and $a_1 \approx_{n,\mathtt{cond}} a_2$ by $t_1 \approx_{n,\mathtt{cond}} \mathtt{cond}(a_2, f_2)(0)$ with $n \geq 0$. Then, by the induction hypothesis, there exists $a'_1 \in \mathtt{GTC}$ such that $a_1 \Rightarrow_n a'_1 \approx_{n,\mathtt{cond}} a'_2$. This case is shown by taking a'_1 as t'_1, since $\mathtt{cond}(a_1, f_1)(0) \Rightarrow_n a'_1 \approx_{n,\mathtt{cond}} a'_2$.

The case of $(\mathtt{cond}\,\mathtt{S})$ is also shown by the induction hypothesis in a similar way to the case $(\mathtt{cond}\,0)$.

The case of (β), namely $t_2 = (\lambda x^A.u_2)v_2 \Rightarrow_n u'_2[v'_2/x^A]$ that is obtained from $u_2 \Rightarrow_n u'_2$ and $v_2 \Rightarrow_n v'_2$. There are u_1, v_1 such that $t_1 = (\lambda x^A.u_1)v_1$, $u_1 \approx_{n,\mathtt{cond}} u_2$, and $v_1 \approx_{n,\mathtt{cond}} v_2$ by $t_1 \approx_{n,\mathtt{cond}} (\lambda x^A.u_2)v_2$ with $n \geq 0$. Then, by the induction hypothesis, there exist $u'_1, v'_1 \in \mathtt{GTC}$ such that $u_1 \Rightarrow_n u'_1 \approx_{n,\mathtt{cond}} u'_2$ and $v_1 \Rightarrow_n v'_1 \approx_{n,\mathtt{cond}} v'_2$. Hence this case is shown by taking $u'_1[v'_1/x^A]$ as t'_1, since $(\lambda x^A.u_1)v_1 \Rightarrow_n u'_1[v'_1/x^A] \approx_{n,\mathtt{cond}} u'_2[v'_2/x^A]$ by Lemma 7.5.

Therefore we obtain the expected result.

The n-safe parallel reduction satisfies the following "pentagon shape" property.

Lemma 7.10 (Pentagon property) *Let $t \in \mathtt{GTC}$. If $t_1 \Leftarrow_n t \Rightarrow_n t_2$, then there exists $t'_1, t'_2 \in \mathtt{GTC}$ such that $t_1 \Rightarrow_n t'_1 \approx_{n,\mathtt{cond}} t'_2 \Leftarrow_n t_2$.*

Proof The lemma is shown by induction on $t \Rightarrow_n t_1$.

The case of (id), namely $t = t_1$, is shown by taking t_2 as both t'_1 and t'_2.

The case of $(\mathtt{S})$, namely $t = \mathtt{S}(u) \Rightarrow_n \mathtt{S}(u_1) = t_1$ that is obtained from $u \Rightarrow_n u_1$. Then $t = \mathtt{S}(u) \Rightarrow_n t_2$ is (id), namely $t_2 = t$, or $(\mathtt{S})$, namely $t_2 = \mathtt{S}(u_2)$ with $u \Rightarrow_n u_2$. The subcase of (id) is immediately shown by taking t_1 as both t'_1 and t'_2. We show the subcase of $(\mathtt{S})$. By the induction hypothesis, there are $u'_1, u'_2 \in \mathtt{GTC}$ such that $u_1 \Rightarrow_n u'_1 \approx_{n,\mathtt{cond}} u'_2 \Leftarrow_n u_2$. Hence this case is shown by taking $(\mathtt{S}(u'_1), \mathtt{S}(u'_2))$ as (t'_1, t'_2), since $t_1 = \mathtt{S}(u_1) \Rightarrow_n \mathtt{S}(u'_1) \approx_{n,\mathtt{cond}} \mathtt{S}(u'_2) \Leftarrow_n \mathtt{S}(u_2) = t_2$.

The cases of (λ) and $(\mathtt{cond})$ are shown similarly by using the induction hypothesis.

The case of $(\mathtt{cond}\,0)$, namely $t = \mathtt{cond}(a, f)(0) \Rightarrow_n t_1$ that is obtained from $a \Rightarrow_n t_1$. Then $t \Rightarrow_n t_2$ is (i) $t = \mathtt{cond}(a, f)(0) \Rightarrow_n \mathtt{cond}(a_2, f_2)(0) = t_2$ with $a \Rightarrow_n a_2$ and $f \Rightarrow_n f_2$, which is (id) or (ap), or (ii) $t = \mathtt{cond}(a, f)(0) \Rightarrow_n t_2$ with $a \Rightarrow_n t_2$, which is $(\mathtt{cond}\,0)$. We first show the subcase (i). By the induction hypothesis, there are $a'_1, a'_2 \in \mathtt{GTC}$ such that $t_1 \Rightarrow_n a'_1 \approx_{n,\mathtt{cond}} a'_2 \Leftarrow_n a_2$. Then the subcase (i) is shown by taking (a'_1, a'_2) as (t'_1, t'_2). Next we show the subcase (ii). By the induction hypothesis, there are $a'_1, a'_2 \in \mathtt{GTC}$ such that $t_1 \Rightarrow_n a'_1 \approx_{n,\mathtt{cond}} a'_2 \Leftarrow_n t_2$. Then the subcase (ii) is shown by taking (a'_1, a'_2) as (t'_1, t'_2).

The case of $(\mathtt{cond}\,\mathtt{S})$ is also shown by the induction hypothesis in a similar way to the case $(\mathtt{cond}\,0)$.

The case of (β), namely $t = (\lambda x^A.u)v \Rightarrow_n u_1[v_1/x^A] = t_1$ that is obtained from $u \Rightarrow_n u_1$ and $v \Rightarrow_n v_1$. Then $t \Rightarrow_n t_2$ is: (i) $t = (\lambda x^A.u)v \Rightarrow_n (\lambda x^A.u_2)v_2 = t_2$ with $u \Rightarrow_n u_2$ and $v \Rightarrow_n v_2$ that is (id) or (ap), or (ii) $t = (\lambda x^A.u)v \Rightarrow_n u_2[v_2/x^A] = t_2$ with $u \Rightarrow_n u_2$ and $v \Rightarrow_n v_2$ that is (λ). The first subcase (i) is shown by the induction hypothesis. We show the second subcase (ii). By the induction hypothesis, there are $u'_1, u'_2, v'_1, v'_2 \in \mathtt{GTC}$ such that $u_1 \Rightarrow_n u'_1 \approx_{n,\mathtt{cond}} u'_2 \Leftarrow_n u_2$ and $v_1 \Rightarrow_n v'_1 \approx_{n,\mathtt{cond}} v'_2 \Leftarrow_n v_2$. Then we have $u_1[v_1/x^A] \Rightarrow_n u'_1[v'_1/x^A] \approx_{n,\mathtt{cond}} u'_2[v'_2/x^A] \Leftarrow_n u_2[v_2/x^A]$ by Lemma 7.8.4. and Lemma 7.5. Then the subcase (ii) is shown by taking $(u'_1[v'_1/x^A], u'_2[v'_2/x^A])$ as (t'_1, t'_2).

Finally we check the case of (ap). The reduction $t \Rightarrow_n t_2$ is (id), (ap), $(\mathtt{cond}\,0)$, $(\mathtt{cond}\,\mathtt{S})$, or (β). The subcases (id) and (ap) are shown by the induction hypothesis. The other subcases are already checked in the above cases.

For $k \geq 0$, we write $t \Rightarrow_n^k t'$ if and only if there is a k-step safe parallel reduction sequence $t = t_0 \Rightarrow_n t_1 \Rightarrow_n \cdots \Rightarrow_n t_k = t'$. The pentagon property also holds for the k-step safe parallel reduction.

Lemma 7.11 (Many step pentagon property) *Let $t \in \mathtt{GTC}$ and $n, k, k_1, k_2 \geq 0$.*

1. *If $t_1 \Leftarrow_n t \Rightarrow_n^k t_2$, then there exist $u_1', u_2' \in \mathtt{GTC}$ such that $t_1 \Rightarrow_n^k t_1' \approx_{n,\mathtt{cond}} t_2' \Leftarrow_n t_2$.*
2. *If $t_1 \Leftarrow_n^{k_1} t \Rightarrow_n^{k_2} t_2$, then there exist $t_1', t_2' \in \mathtt{GTC}$ such that $t_1 \Rightarrow_n^{k_2} t_1' \approx_{n,\mathtt{cond}} t_2' \Leftarrow_n^{k_1} t_2$.*

Proof The point 1. is shown by induction on k. The base case $k = 0$ is immediately shown by taking t_1 as both t_1' and t_2'. We show the case $k > 0$. Let $t \Rightarrow_n^{k-1} t_* \Rightarrow_n t_2$. By the induction hypothesis, there are t_1' and t_*' such that $t_1 \Rightarrow_n^{k-1} t_1' \approx_{n,\mathtt{cond}} t_*' \Leftarrow_n t_*$. Then, by $t_*' \Leftarrow_n t_* \Rightarrow_n t_2$ and the pentagon property, we have $t_*' \Rightarrow_n t_*'' \approx_{n,\mathtt{cond}} t_2'' \Leftarrow_n t_2$ for some t_*'' and t_2''. Hence, by $t_1' \approx_{n,\mathtt{cond}} t_*' \Rightarrow_n t_*''$ and Lemma 7.9, we have $t_1' \Rightarrow_n t_1'' \approx_{n,\mathtt{cond}} t_*''$ for some t_1''. Therefore, using the transitivity of $\approx_{n,\mathtt{cond}}$, we obtain $t_1 \Rightarrow_n^k t_1'' \approx_{n,\mathtt{cond}} t_2'' \Leftarrow_n t_2$ as we wished.

The point 2. is shown by induction on k_1. The base case $k_1 = 0$ is immediately shown by taking t_2 as both t_1' and t_2'. To show the inductive case $k > 0$, let $t \Rightarrow_n^{k_1-1} t_* \Rightarrow_n t_1$. Then, by the induction hypothesis, $t_* \Rightarrow_n^{k_2} t_*' \approx_{n,\mathtt{cond}} t_2' \Leftarrow_n^{k_1-1} t_2$ holds for some t_*' and t_2'. By $t_1 \Leftarrow_n t_* \Rightarrow_n^{k_2} t_*'$ and the point 1., we have $t_1 \Rightarrow_n^{k_2} t_1'' \approx_{n,\mathtt{cond}} t_*'' \Leftarrow_n t_*'$ for some t_1'' and t_*''. Then, by $t_2' \approx_{n,\mathtt{cond}} t_*' \Rightarrow_n t_*''$ and Lemma 7.9, $t_2' \Rightarrow_n t_2'' \approx_{n,\mathtt{cond}} t_*''$ for some t_2''. Hence, using the transitivity of $\approx_{n,\mathtt{cond}}$, we have $t_1 \Rightarrow_n^{k_2} t_1'' \approx_{n,\mathtt{cond}} t_2'' \Leftarrow_n^{k_1} t_2$ as we wished.

Proof of the weak Church-Rosser of n-safe reduction.

Assume that $t \in \mathtt{GTC}$ and $t_1 \leftarrow\!\!\!\shortmid^*_{n,\mathtt{safe}} t \mapsto^*_{n,\mathtt{safe}} t_2$. By Lemma 7.8.2., we have $t_1 \Leftarrow_n^{k_1} t \Rightarrow_n^{k_2} t_2$ for some k_1 and k_2. Then there exist $t_1', t_2' \in \mathtt{GTC}$ such that $t_1 \Rightarrow_n^{k_2} t_1' \approx_{n,\mathtt{cond}} t_2' \Leftarrow_n^{k_1} t_2$. By Lemma 7.8.3., $t_1 \mapsto^*_{n,\mathtt{safe}} t_1' \approx_{n,\mathtt{cond}} t_2' \leftarrow\!\!\!\shortmid^*_{n,\mathtt{safe}} t_2$ holds as we wished.

8 Terms with the Global Trace Condition are Finite for Safe Reductions

In this section we prove that for all $n \in N$, every infinite reduction sequence π from some term of GTC includes only finitely many "0-safe" (safe) reduction steps: from some point on, all reduction steps are unsafe.

From this, we will prove that in GTC the reduction sequences made of only "n-safe" reductions from some point on stop, therefore are finite. Namely, this implies strong normalization for "n-safe" reduction steps: no matter how we reduce within the safe level of a term, eventually no reductions are left, therefore we obtained some safe normal form.

We introduce the property of being "finite for n-safe reductions".

Definition 8.1 *Assume $t \in$ WTyped. We say that t is finite for n-safe reductions if and only if all infinite reduction paths include only finitely many n-safe reductions (in all infinite reduction paths, from some point on all reductions are not n-safe).*

In the following, we explicitly write $t[x_1, \ldots, x_n]$, when each free variable in a term t is some x_i, and, under this notation, we also write $t[a_1, \ldots, a_n]$ for $t[a_1/x_1, \ldots, a_n/x_n]$.

It is enough to consider the case of 0-safe reductions, the general case will follow. As we previously said, we abbreviate "0-safe" with "safe". We have to prove that all terms in GTC are finite for safe reductions. For, we define total well-typed terms by induction on the type, as in Tait's normalization proof.

Definition 8.2 (Total well-typed terms of $\mathbb{A}$) *Let $t \in$ WTyped and $t : T$. We define "t is total of type T" by induction on T*

1. *Let T be any atomic type: $T = N$ or $= \alpha$ for some type variable α. Then t is total of type T if and only if t is finite for safe reductions.*
2. *Let T be any arrow type $A \to B$. Then f is total of type T if and only if for all total a of type A we have $f(a)$ total of type B.*

A term $t[\vec{x}]$ with free variables $\vec{x} : \vec{B}$ is total by substitution if and only if: $t[\vec{v}]$ is total for all totals $\vec{x} : \vec{B}$.

Let U be an atomic type, $t[\vec{x}] : \vec{A} \to U$, and $\vec{x} : \vec{B}$. A total assignment of $t[\vec{x}]$ is $(\vec{v}, \vec{a})$, where $\vec{v} : \vec{B}$ and $\vec{a} : \vec{A}$ are total terms, and $t[\vec{v}](\vec{a})$ is total.

By definition, any total term is well-typed, in particular it has exactly one type.

We define a well-founded relation predecessor relation on terms finite for safe reductions and of type N. We recall that $\mapsto$ denotes one reduction step (contraction of a single redex which is a subterm), $\mapsto^*$ denotes zero or more reduction steps and $\mapsto^+$ denotes one or more reduction steps.

Definition 8.3 (The S-order) *Assume $t, u \in$ WTyped and $t, u : N$ (possibly open terms). Then:*

$$(u \prec t) \Leftrightarrow (t \mapsto^* \mathtt{S}(u))$$

Example 8.1 *If $t = \mathtt{S}^2(x)$ then $x \prec \mathtt{S}(x) \prec t$. If $t = \mathtt{S}(t)$ then $t \prec t$ and $\prec$ is not well-founded on t:*

We did not prove Church-Rosser yet, therefore we ignore whether all decreasing sequences of $\prec$ have the same length.

However, we can prove that if t is a term finite for safe reductions, then any $\prec$-decreasing sequence from t terminates.

Lemma 8.4 (The $\prec$-order) *Assume $t \in$ WTyped has type N, and t is finite for safe reductions .*

1. *There is no infinite sequence $\sigma : t = t_0 \mapsto^* \mathtt{S}(t_1) \mapsto^* \mathtt{S}^2(t_2) \mapsto^* \ldots$*
2. *$\prec$ is well-founded on total terms of type N.*

Proof 1. Assume that there is such σ to show a contradiction. Since t is finite for safe-reductions, σ only has finitely many safe-reductions. Thus, from some $k \in N$ there are no more safe reductions from $\mathtt{S}^k(t_k)$. This implies that for some $\mathtt{cond}$-free term u and some terms $f_1, g_1, \ldots, f_m, g_m$ we have $t_k = u[\mathtt{cond}(f_1, g_1), \ldots, \mathtt{cond}(f_m, g_m)]$ and all reductions from t_k on are inside some $g_1, \ldots, g_m$. This means for all $h \in N$, $h \geq k$ we have $\mathtt{S}^h(t_h) = u[\mathtt{cond}(f_1, g'_1), \ldots, \mathtt{cond}(f_n, g'_m)]$ for some $g'_1, \ldots, g'_m$. This implies that first h symbols of u are $\mathtt{S}$. This is a contradiction when h is larger than k, which is the number of symbols in u.

2. Assume for contradiction that there is some infinite sequence $\ldots t_n \prec \ldots \prec t_2 \prec t_1 \prec t_0$ from some $t_0 : N$ total. By definition, t_0 is finite for safe reductions. Then there is some infinite sequence $\sigma : t = u_0 \mapsto^* \mathtt{S}(u_1) \mapsto^* \mathtt{S}^2(u_2) \mapsto^* \ldots$, contradicting point 1. above.

We will prove that if t is not total, then we can assign total terms to the sub-terms of t in an infinite path of a proof $\Pi : \Gamma \vdash t : A$ in a way compatible with traces.

Definition 8.5 (Trace-compatible Assignment) *Assume $\pi = (\Gamma_1 \vdash t_1 : A_1, \ldots, \Gamma_n \vdash t_n : A_n, \ldots)$ is any finite or infinite branch of a typing proof Π and $\vec{v} = (r_1, \ldots, r_n, \ldots)$ is any sequence of total assignments, where, for each i, $r_i = (r_{i,1}, \ldots, r_{i,m_i})$ is one for t_i. $\vec{v}$ is trace-compatible in an index i of π if and only if it satisfies the following condition:*

for all j index of an N-argument of t_i, all k index of an N-argument of t_{i+1}, if j is connected to k then:

1. *if j progresses to k then $r_{i+1,k} \prec r_{i,j}$*
2. *if j does not progress to k then $r_{i+1,k} = r_{i,j}$.*

$\vec{v}$ is trace-compatible if it is trace-compatible in all i.

Now we will prove that if an infinite branch π of a proof tree $\Pi : \Gamma \vdash t : A$ has a trace-compatible assignment made of total terms, then all traces σ of π progress only finitely many times and the term t is not in $\mathtt{GTC}$ (t does not satisfy the global trace condition).

Proposition 8.6 (Trace assignment) *Assume $\Pi : \Gamma \vdash t : A$ and there is some infinite path π of Π for which we have some total trace-compatible assignment ρ to π.*

1. *Any trace σ in π progresses only finitely many times.*
2. *$t \notin \mathtt{GTC}$.*

Proof 1. By definition of trace-compatible assignment, if at step $i \in N$ the trace σ progresses, then $\sigma(i+1) \prec \sigma(i)$, and if σ does not progress, then $\sigma(i+1) = \sigma(i)$ The assignment is made of total terms, therefore $\prec$ is well-founded by lemma 8.4.2.. Thus, any trace in π progresses only finitely many times, as we wished to show.

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

We now continue with our Tait's-style proof of Strong Normalization. We check that total terms are closed by reductions, by application and by variables. Any total term is finite for safe reductions.

Lemma 8.7 *Assume $t, u, f, a \in \mathtt{WTyped}$, $n \in N$ and A, B, T are types.*

1. *Let $t : A$ and $t \mapsto^* u$. If t is total, then u is total.*
2. *If $f : A \to B$ and $a : A$ are total terms, then $f(a)$ is total.*
3. *$x^T : T$ is total.*
4. *If t is total then t is finite for safe reductions.*
5. *Let U be any atomic type, and $t[\vec{x}] : \vec{A} \to U$ be a term whose all free variables are $\vec{x} : \vec{B}$.*
   If for all total $\vec{u} : \vec{B}$, $\vec{a} : \vec{A}$ the term $t[\vec{u}]\vec{a} : U$ is finite for safe reductions, then the term $t[\vec{x}]$ is total by substitution.

Proof 1. We show *point 1.* by induction on A. We assume that $t : A$ and $t \mapsto^* u$. and t is total, in order to prove that u is total. By the subject reduction property, u has type A.

(1) We show the *base case*, namely when $A = N, \alpha$ is an atomic type. By the assumption, t is total. By definition of t total for T atomic, all infinite reductions from t only include finitely many safe reductions, for all $n \in N$. In particular, all infinite reductions $\sigma : t \mapsto \ldots \mapsto u \mapsto u_1 \mapsto u_2 \ldots$ passing through u only include finitely many safe reductions. We conclude that all infinite reductions $\sigma' : u \mapsto u_1 \mapsto u_2 \ldots$ from u only include finitely many safe reductions. From $u : A = N, \alpha$ we conclude that u is total.

(2) We show the *induction case*, namely when $A = (A_1 \to A_2)$. Take any arbitrary total term $a : A_1$ in order to prove that $u(a) : A_2$ is total. Then we have $t(a) \mapsto^* u(a)$ and $t(a) : A_2$ is total by the assumption that t is total. Hence $u(a)$ is total by $t(a) \mapsto^* u(a)$ and the induction hypothesis on A_2. We conclude that $u : A_1 \to A_2$ is total.

2. If $f : A \to B$, $a : A$ are total terms, then $f(a)$ is total by definition of total.

3. We prove that $x^T : T$ is total. We actually prove a little more, that for all total $\vec{a} : \vec{A}$, if $T = \vec{A} \to U$ then $x(\vec{a}) : U$ is total. The thesis follows if we take $\vec{a} = \texttt{nil}$. We argue by induction on U.

(1) Assume U is atomic. Then every reduction on $x(\vec{a}) : U$ takes place in $\vec{a}$. By definition of total for an atomic type we have to prove that in all infinite reduction sequences from $x(\vec{a}) : U$ there are only finitely many safe reduction. All reductions on $x(\vec{a})$ take place on $\vec{a}$, and since each a_i in $\vec{a}$ is total only finitely many safe reductions are possible, as we wished.

(2) Assume $U = A_1 \to A_2$. By definition of total for an arrow type we have to prove that for all total a we have $x(\vec{a}, a) : A_2$ total. This follows by induction hypothesis on A_2.

4. *We assume that $t : U$ is total in order to prove that t is finite for safe reductions*, for all $n \in N$. We argue by induction on U.

(1) If U is atomic then the thesis is true by definition of total.

(2) Suppose $U = (A_1 \to A_2)$. By point 5. above, $x^{A_1} : A_1$ is total, therefore $t(x) : A_2$ is total and by induction hypothesis on A_2 any infinite reduction sequence from $t(x)$ only includes finitely many safe reductions. Any infinite reduction sequence $\sigma : t = t_0 \mapsto_1 t_1 \mapsto_1 t_2 \mapsto_1 \ldots$ from t can be raised to an infinite reduction sequence $\tau : t(x) = t_0(x) \mapsto_1 t_1(x) \mapsto_1 t_2(x) \mapsto_1 \ldots$ from $t(x)$ while preserving the fact that a reduction is safe, because t occurs in no `cond` in $t(x)$. We conclude that σ only includes finitely many safe reductions.

5. We show that $t[\vec{u}]$ is total by induction on the number $|\vec{A}|$ of elements of $\vec{A}$.

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

(2) We show the *induction case*. Let $\vec{A} = A_0, \vec{A'}$. Take arbitrary totals $\vec{u} : \vec{B}$, $\vec{a'} : \vec{A'}$, and $a_0 : A_0$. By the assumption, we have that $t[\vec{u}]a_0\vec{a'} : U$ is total for all vector of total terms $\vec{a'}$. Then $t[\vec{u}]a_0 : \vec{A'} \to U$ is total for all total $a_0 : A_0$ by the induction hypothesis on $\vec{A'} \to U$. By definition of total we deduce that $t[\vec{u}] : A_0, \vec{A'} \to U$ is total.

We conclude that $t[\vec{x}]$ is total by substitution, as we wished to show.

Let $\Pi = (T, \phi)$ be a proof of $\Gamma \vdash t : A$ and l_n be a node of Π, that is, a list of integers $l_n = (e_1, \ldots, e_{n-1}) \in |\Pi|$ for some $n \in N$. We write $\Gamma_n \vdash t_n : A_n = \texttt{Lbl}(\Pi, l_n)$ for the sequent labelling the node l_n. We want to prove that all terms of `GTC` are total by substitution. If we consider the substitution of a variable with itself (a variable is a total term by 8.7.5.), we will deduce that all all terms of `GTC` are total, hence finite for safe reductions, hence strongly normalizing for safe reductions.

One problem in the proof of the theorem is that reduction does not commute with the second argument of substitution. That is, if $b \mapsto^* a$ we cannot deduce that $v[b] \mapsto^* v[a]$. The reason is that we could have infinitely many free occurrences of x in v, and it could take an infinite number of steps to reduce to a each b which has been replaced to x in v.

However, we can prove a weaker property: if there is some infinite reduction from $v[a]$, then there is some infinite reduction from $v[b]$, such that if the first reduction has infinitely many safe reduction steps, then the second reduction has infinitely many safe reduction steps, too.

To this aim, we first define a simulation relation from reductions from $v[a]$ to reductions from $v[b]$.

We recall that $\mapsto$, $\mapsto^*$, $\mapsto^+$ denote respectively one, zero or more, one or more reduction steps.

Lemma 8.8 (Safety-preserving Simulation) *Define a binary relation $R \subseteq \mathbb{\Lambda} \times \mathbb{\Lambda}$ by: tRu if and only if there are $v, a, b \in \mathbb{\Lambda}$ and a variable x such that:*

$$(b \mapsto^* a) \quad \textit{and} \quad (t = v[a/x]) \quad \textit{and} \quad (u = v[b/x])$$

Then R is a safety-preserving simulation on $\mathbb{\Lambda}$ between $\mapsto$ and $\mapsto^+$, namely:

1. *whenever $t, u, t' \in \mathbb{\Lambda}$, tRu and $t \mapsto t'$ then for some $u' \in \mathbb{\Lambda}$ we have $t'Ru'$ and $u \mapsto^+ u'$.*
2. *Besides, if $t \mapsto t'$ is safe then some step in $u \mapsto^+ u'$ is safe, too.*

Proof Assume that $t \in \mathbb{\Lambda}$, $u \in \mathbb{\Lambda}$, $t' \in \mathbb{\Lambda}$ and tRu and $t \mapsto t'$. We have to prove that for some $u' \in \mathbb{\Lambda}$ we have $t'Ru'$ and $u \mapsto^+ u'$ and besides that if $t \mapsto t'$ is safe then some step in $u \mapsto^+ u'$ is safe, too.

The assumption tRu unfolds to: for some $a, b, w \subseteq \mathbb{\Lambda}$, some variable y, we have $(b \mapsto^+ a)$ and $(t = w[a/y])$ and $(u = w[b/y])$. By renaming y in v with some variable $x \notin \mathrm{FV}(w, a, b)$ we have that $(t = w[x/y][a/x])$ and $(u = w[x/y][b/x])$ and $x \notin \mathrm{FV}(w, a, b)$. If we set $v = w[x/y]$ we have $(t = v[a/x])$ and $(u = v[b/x])$ and $x \notin \mathrm{FV}(a, b)$.

The assumption $t \mapsto t'$ implies that for some unique redex $r \sqsubseteq t$, for some context $C[\cdot]$ we have $t = C[r]$ and $t' = C[r']$ and $r \mapsto r'$. The possible shapes of r are $r = (\lambda x.c)(d), \mathtt{cond}(f, g)(0), \mathtt{cond}(f, g)(\mathtt{S}(e))$ and the corresponding shapes of r' are $r' = c[d/x], f, g(e)$. Thus, for some r_1, r_2 we have $r = r_1(r_2)$. Assume π is the position of r in t. If $r \mapsto r'$ is safe, then π crosses no right-hand side of any `cond`. We argue by cases on π. Either π is some position of v or it is not.

1. *Assume that π is not the position of a node of v.* Then there is some free occurrence of x in v, with position θ, such that $\pi \geq \theta$. Assume z is the term obtained by replacing this occurrence of x with a single variable $x_0 \notin \mathrm{FV}(v)$. Then $v = z[x/x_0]$, therefore $t = z[x/x_0][a/x]$ and $u = z[x/x_0][a/x]$. We deduce $t = z[a/x_0, a/x] =$ (by $x \notin \mathrm{FV}(a, b)$) $= z[a/x_0][a/x]$ and $u = z[b/x_0, b/x] =$ (by $x \notin \mathrm{FV}(a, b)$) $= z[b/x_0][b/x]$. Let us abbreviate $D[\cdot] = z[\cdot/x_0]$, with D context: then $t = D[a][a/x]$ and $ut = D[b][b/x]$. For some context C we have $a = C[r] \mapsto C[r']$. From $x \notin \mathrm{FV}(a) \cup \mathrm{FV}(b)$ we deduce that there is some context D such that: $t = D[a][a/x] = D[C[r]][a/x]$ and $u = D[b][b/x]$ and $t' = D[C[r']][a/x]$. We choose $u' = D[C[r']][b/x]$. We deduce $t'Ru'$. From $r \mapsto r'$ we deduce $D[r] \mapsto D[r']$, then $b \mapsto^* a = C[r] \mapsto C[r']$, hence $b \mapsto^+ C[r']$. Eventually we deduce $u = D[b][b/x] \mapsto^* D[C[r]][b/x] \mapsto D[C[r']][b/x] = u'$. If the reduction $t = D[C[r]][a/x] \mapsto D[C[r']][a/x]$ is safe, then the last reduction$D[C[r]][b/x] \mapsto D[C[r']][b/x]$ in $u \mapsto u'$ is safe.
2. *Assume that π the position of some node s in v and s is some redex.* There exists some context D such that $v = D[s]$. Then $r = s[a/x]$ and $r' = s'[a/x]$ for some redex $s = s_1(s_2)$ of v contracted to s' with the same reduction used for r. We deduce that: $t = v[a/x] = D[s][a/x]$ and $u = v[b/x] = D[s][b/x]$ and $t' = D[s'][a/x]$. We choose $u' = D[s'][b/x]$. From $b \mapsto^* a$ we deduce that $t' = D[s'][a/x]$ and $u' = D[s'][b/x]$ are related by R. Thus, we have $t'Ru'$. From $s \mapsto s'$ we deduce that $u = D[s][b/x] \mapsto D[s'][vecb/x] = u'$. If $r \mapsto r'$ is safe, then π crosses no right-hand side of any `cond`, therefore the reduction $s \mapsto s'$ is safe in u.
3. *Assume that π the position of some node s in v, yet s is no redex.* There exists some context D such that $v = D[s]$. π is the position of an application $r_1(r_2)$ in t, therefore π is the position of some application $s = s_1(s_2)$ is v. If s is no redex, then either s_1, r_1 do not start with the same symbol, or and s_2, r_2 do not start with the same symbol. In the first sub-case we have $s_1 = x$, $r_1 = s_1[a/x] = a$ and $r = r_1(r_2) = a(r_2)$. In the second sub-case we have $s_2 = x$, $r_2 = s_2[a/x] = a$ and $r = r_1(r_2) = r_1(a)$. We cannot have both case at the same time, because the type of r_2 is the type of an argument of r_1. Therefore r_1, r_2 have two different types and a cannot have both types. Thus, if $s_1 = x$ then r_2, s_2 start with the same symbol, and if $s_2 = x$ then r_1, s_1 start with the same symbol.
   (1) *Sub-case: $s_1 = x$ and r_2, s_2 start with the same symbol.* We have $r1(r_2) = a(s_2[a/x]) =$ (by $x \notin \mathrm{FV}(a)$) $a(s_2)[a/x]$ If we re-define $s_1 = a$ we have that s_1 has the same starting symbol as r_1 and s_2 has the same starting symbol as r_2. Thus, $s = s_1(s_2)$ is a redex, it reduces to some s' with the same reduction we have in $r \mapsto r'$, and by unicity of contractum we have $r' = s'[a/x]$. For the context D such that $v = D[s]$ we have: $t = v[a/x] = D[s][a/x] = D[a(s_2)][a/x]$ and $u = D[b(s_2)][b/x]$ and $t' = D[s'][a/x]$. We choose $u' = D[s'][b/x]$ and we deduce $t'Ru'$. From $b(s_2) \mapsto^* a(s_2) = s \mapsto s'$ we deduce that $u = D[b(s_2)][b/x] \mapsto^* D[s][b/x] \mapsto D[s'][b/x] = u'$.

(2) *Sub-case:* $s_2 = x$ *and* r_1, s_1 *start with the same symbol.* We have $r = r_1(r_2) = r_1(a)$ and $r_1 = s_1[a/x]$ for some s_1 with the same starting symbol as r_1. We re-define $s_2 = a$. Then $s_2 = a$ has the same starting symbol as $r_2 = a$, therefore $s = s_1(s_2) = s_1(a)$ reduces to some s' such that $r' = s'[a/x]$ For some context D we have: $t = D[s_1(a)][a/x]$ and $u = D[s_1(b)][b/x]$ and $t' = D[s'][a/x]$. We choose $u' = D[s'][b/x]$ and we deduce $t'Ru'$. From $s_1(b) \mapsto^* s_1(a) = s \mapsto s'$ we deduce that $u = D[s_1(b)][b/x] \mapsto^* D[s_1(a)][b/x] = D[s_1(s_2)][b/x] \mapsto D[s'][b/x] = u'$.

In all cases, if $t \mapsto t'$ is safe then the reduction $s[a/x] \mapsto s'[a/x]$ is safe in $D[s][a/x] \mapsto D[s'][a/x] = t'$ and therefore the reduction $s[b/x] \mapsto s'[b/x]$ is the last reduction step in $u \mapsto D[s'][b/x] = u'$ and it is safe, too.

Now we can prove the required property for reductions with infinitely many safe reduction steps.

Lemma 8.9 (Safe infinite reductions and Substitution) *Let $a \in \mathbb{A}$, $b \in \mathbb{A}$ and $v \in \mathbb{A}$. If $b \mapsto^+ a$ and there is some reduction with with infinitely many safe steps from $v[a/x]$, then there is some reduction with with infinitely many safe steps from $v[b/x]$.*

The Lemma above will be enough to prove our Main Theorem.

Theorem 8.10 (Main Theorem) *Assume $\Pi : \Gamma \vdash t : A$ (hence $t \in$ `WTyped`). If t is not total by substitution, then $t \notin$ `GTC`, i.e.: there is some infinite path $\pi = (e_1, e_2, \ldots)$ of Π with no infinite progressing trace.*

Proof Assume that t is not total by substitution. Let $\vec{x} : \vec{D}$ and $\vec{A} \to U$ for some *atomic* U be Γ and A, respectively. By Proposition 8.6.2. it is enough to prove that Π has some infinite branch $\pi = (e_1, e_2, \ldots)$ and some *total* trace-compatible assignment ρ for π. By case 5. of Lemma 8.7, there exist total terms $\vec{a} : \vec{A}$ and $\vec{d} : \vec{D}$ for which there is an infinite reduction σ from $t[\vec{d}]\vec{a}$ having infinitely many safe reductions. By induction on $i \in \mathbb{N}$, for each i, we construct a path $l_i = (e_1, \ldots, e_{i-1}) \in |\Pi|$ and a total assignment $\vec{v_i} = (\vec{d_i}, \vec{a_i})$ such that $(\vec{v_1}, \ldots, \vec{v_i})$ is a total trace-compatible assignment for the node l_i.

We recall that "trace compatible in i" means: if $i-1$ is a progress point, namely if $t_{i-1} = \mathtt{cond}(f, g)$ and $e_i = 2$ and $t_i = g$, then $\vec{a_{i-1}} = a', \vec{a'}$ and a', the first unnamed argument of $\mathtt{cond}(f, g) : N \to A$, reduces to $\mathtt{S}(a'')$ while $\vec{a_i} = a'', \vec{a'}$. In all other cases two corresponding arguments are equal.

We first define $(l_1, \vec{d_1}, \vec{a_1})$ for the root node t of Π. In this case $l_1 = \mathtt{nil}$ and $(\vec{d}, \vec{a})$ are total terms such that $t[\vec{d}](\vec{a})$ is not total. Trace compatibility is vacuous because the branch l_1 does not contain two nodes.

Next, assume that $(l_i, \vec{d_i}, \vec{a_i})$ is already constructed. Then we define $(l_{i+1}, \vec{d_{i+1}}, \vec{a_{i+1}})$ by the case analysis on the last rule for the node l_i in Π.

1. The case of `weak`, namely $\mathtt{Lbl}(\Pi, l_{i+1}) = \Gamma \vdash t_{i+1} : \vec{A_i} \to U$ is obtained from the induction hypothesis for $\Gamma' \vdash t_i : \vec{A_i} \to U$. We have:
   (1) $t_i = t_{i+1}$.
   (2) $\Gamma = x_1 : D_1, \ldots, x_n : D_n$, $\Gamma' = x'_1 : D'_1, \ldots, x'_m : D'_m$, and $\Gamma \subsetneq \Gamma'$ with an injection $\phi : \{1, \ldots, n\} \to \{1, \ldots, m\}$ between contexts.
   (3) $x_i = x'_{\phi(i)}$ and $D_i = D'_{\phi(i)}$ for all $i \in \{1, \ldots, n\}$.

   By the induction hypothesis, $t_i[d_{i,\phi(1)}/x_1, \ldots, d_{i,\phi(n)}/x_n]\vec{a_i} = t_i[d_{i,1}/x'_1, \ldots, d_{i,m}/x'_m]\vec{a_i}$ is not total, where $\vec{d_i} = d_{i,1} \ldots d_{i,m}$. Then we define $l_{i+1} = l_i \star (1)$ taking the unique child node of l_i in Π, and we also define $\vec{d_{i+1}}$ and $\vec{a_{i+1}}$ by $d_{i,\phi(1)} \ldots d_{i,\phi(n)}$ and $\vec{a_i}$, respectively. This is an assignment with total terms. $((\vec{d_i}, \vec{a_i}), (\vec{d_{i+1}}, \vec{a_{i+1}}))$ is trace compatible for (t_i, t_{i+1}): if two arguments are connected then they are assigned to the same term.
2. The case of `var`-rule, namely $\mathtt{Lbl}(\Pi, l_i) = \Gamma \vdash x : D$ for some $x_{i,k} : D_{i,k} = x : D \in \Gamma$ cannot be, because $t_i[\vec{d_i}/\vec{x_i}] = d_{i,k}$ is total by assumption on $\vec{d}$.
3. The case of 0-rule, namely $\mathtt{Lbl}(\Pi, l_i) = \Gamma \vdash 0 : N$, cannot be. Indeed, $t_i = 0$ is total because 0 is a numeral.

4. The case of S-rule, namely $\mathtt{Lbl}(\Pi, l_i) = \Gamma \vdash t_i : N = \Gamma \vdash \mathtt{S}(u) : N$ for some u is obtained from our assumptions on $\Gamma \vdash u : N$. In this case $\vec{a_i}$ is empty, $t_{i+1} = u$, and by the induction hypothesis $\mathtt{S}(u)[\vec{d_i}] : N$ has an infinite reduction with infinitely many safe reductions $\mathtt{S}(u) \mapsto \mathtt{S}(u_1) \mapsto \mathtt{S}(u_2) \mapsto \ldots$, all taking place on u. Then, by the definition of total, $u[\vec{d_i}] = t_{i+1}[\vec{d_i}]$ is not total, because the $u \mapsto_1 u_1 \mapsto_1 u_2 \mapsto_1 \ldots$ is an infinite reduction with infinitely many safe reductions.
   We define $e_i = 1$, the index of the unique child node of $l_{i+1} = l_i\star(1)$ in the S-rule, and we also define $\vec{d_{i+1}} = \vec{d_i}$ and $\vec{a_{i+1}} = ()$. This is an assignment with total terms and trace compatible for (t_i, t_{i+1}): if two arguments are connected then they are assigned to the same term.
5. The case of the $\mathtt{ap}_{\_v}$-rule, namely $\mathtt{Lbl}(\Pi, l_i) = \Gamma \vdash t_i : \vec{A} \to U$, with $t_i = f[\vec{x}](u[\vec{x}])$ for some f and u. The premises of the $\mathtt{ap}_{\_v}$-rule are $\Gamma \vdash f[\vec{x}] : B \to \vec{A} \to U$ and $\Gamma \vdash u[\vec{x}] : B$, where u is <u>not</u> a variable. By the induction hypothesis, there is some infinite reduction from $t_i[\vec{d_i}]\vec{a_i} = f[\vec{d_i}](u[\vec{d_i}])\vec{a_i} : U$ with infinitely many safe-reductions. We argue by cases on the statement: $u[\vec{d_i}] : B$ *is total.*

   (1) We first consider the subcase: $u[\vec{d_i}] : B$ *is total.* We define $b = u[\vec{d_i}]$, then $l_{i+1} = l_i\star(1)$, taking the first premise of the rule, and we define $\vec{d_{i+1}} = \vec{d_i}$ and $\vec{a_{i+1}} = b, \vec{a_i}$. This is an assignment with total values, and providing an infinite reduction with infinitely many safe reductions, as expected. The connection from $(\vec{d_i}, \vec{a_i})$ to $(\vec{d_{i+1}}, \vec{a_{i+1}}) = (\vec{d_i}, b, \vec{a_i})$ is trace-compatible: all connected N-argument of $t_i = f(u)[\vec{x}]$ and $t_{i+1}[\vec{x}] = f[\vec{x}]$ are the same, because the only fresh argument of $f[\vec{x}]$ is b and no argument of $f(u)[\vec{x}]$ is connected to it.

   (2) Next we consider the subcase that $u[\vec{d_i}] : B$ *is not total.* Let $B = \vec{C} \to U$. By lemma 8.7.5. there is a sequence of values $\vec{c} : \vec{C}$ and an infinite reductions from $u[\vec{d_i}]\vec{c} : U$ with infinitely many safe reductions. Define $l_{i+1} = l_i\star(2)$ taking the second premise of the rule, and define $\vec{d_{i+1}} = \vec{d_i}$ and $\vec{a_{i+1}} = \vec{c}$. This is an assignment with total terms providing an infinite reductions with infinitely many safe reductions, as expected. The connection from $(\vec{d_i}, \vec{a_i})$ to $(\vec{d_{i+1}}, \vec{a_{i+1}}) = (\vec{d_i}, \vec{c})$ is trace compatible: all connected N-argument of $t_i = f(u)[\vec{x}]$ and $t_{i+1} = u[\vec{x}]$ are in $\vec{d_i}$ and therefore are the same, and no unnamed arguments in $\vec{a_i}$ and $\vec{a_{i+1}}$ are connected each other.
6. The case of $\mathtt{ap}_v$-rule, namely $\mathtt{Lbl}(\Pi, l_i) = \Gamma \vdash f[\vec{x}](x) : \vec{A} \to U$ is obtained from $\Gamma \vdash f[\vec{x}] : D, \vec{A} \to U$, where $\Gamma = x_1 : D_1, \ldots, x_m : D_m$ and $x : D \in \Gamma$, therefore $x : D = x_k : D_k$ for some $k \in [1, m]$. By the induction hypothesis, there is an infinite reduction from $f[\vec{d_i}]d_{i,k}\vec{a_i} : U$ with infinitely many safe reductions, where $\vec{d_i} = (d_{i,1}, \ldots, d_{i,m})$. Define $l_{i+1} = l_i\star(1)$ as the unique child of l_i in Π, and also define $\vec{d_{i+1}} = \vec{d_i}$ and $\vec{a_{i+1}} = d_{i,k}, \vec{a_i}$. This is an assignment with total terms providing an infinite reductions with infinitely many safe reductions, as expected. The connection from $(\vec{d_i}, \vec{a_i})$ to $(\vec{d_{i+1}}, \vec{a_{i+1}}) = (\vec{d_i}, d_{i,k}, \vec{a_i})$ is trace compatible: all connected N-arguments 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, if $D_k = N$, the value $d_{i,k}$ of type N for the variable x_k in $t_i[\vec{x}] = f[x_1, \ldots, x_k, \ldots, x_m](x_k)$ is duplicated to the term $d_{i,k}$ assigned to the first unnamed argument of $f[\vec{x}]$.
7. The case of λ-rule, namely when a sequent $\mathtt{Lbl}(\Pi, l_i) = \Gamma \vdash t_i : A, \vec{A} \to U$ with $t_i = \lambda x^A.u[\vec{x}, x]$ is obtained from $\Gamma, x : A \vdash t_{i+1}[\vec{x}, x^A] : \vec{A} \to U$, where $t_{i+1}[\vec{x}, x] = u[\vec{x}, x]$. By the induction hypothesis we have an infinite reduction sequence σ from $t_i[\vec{d_i}]\vec{a_i} = (\lambda x.(u[\vec{d_i}, x]))\vec{a_i} : U$ with infinitely many safe reductions, where $\vec{a_i} = a, \vec{b}$. Define $l_{i+1} = l_i\star(1)$ as the unique child of l_i in Π, and $\vec{d_{i+1}} = \vec{d_i}, a$ and $\vec{a_{i+1}} = \vec{b}$. This is an assignment with total terms. The connection from $(\vec{d_i}, \vec{a_i})$ to $(\vec{d_{i+1}}, \vec{a_{i+1}}) = (\vec{d_i}, a,\ \vec{b})$ is trace compatible: all connected N-argument of $t_i = \lambda x^A.u[\vec{x}, x]$ and $t_{i+1} = u[\vec{x}, x]$ are the same, except for the first unnamend argument a of $\vec{a_i}$ which is moved to the last named argument of $u[\vec{x}, x]$, with name x.
   We have to prove that there is some infinite reduction from $t_{i+1}[\vec{d_{i+1}}](\vec{a_{i+1}})$ with infinitely many safe reductions. We argue by case.

   (1) Suppose all reductions in the infinite reduction σ are inside $\lambda x.(u[\vec{d_i}, x])$ or $\vec{a_i}$. Then there are finitely many safe reductions in $\vec{a_i}$ because $\vec{a_i}$ is total. Thus, there are infinitely many safe reductions on the part of σ taking place on $\lambda x.(u[\vec{d_i}, x])$. We conclude that there is an infinite reduction from $u[\vec{d_i}, x]$ with infinitely many safe reduction. This is true for $u[\vec{d_i}, a]$, too, because reductions and safe reductions are closed by substitution of x with a.

(2) Suppose there is some reduction in σ contracting the first β-redex. Then $(\lambda x.(u[\vec{d_i}, x]))a\vec{b}$ reduces first to some $(\lambda x.v[x])a'\vec{b'}$, then to $v[a']\vec{b'}$, with: $u[\vec{d_i}, x] \mapsto^* v[x]$ and $a \mapsto^* a'$ and $\vec{b} \mapsto^* \vec{b'}$. Then the reduction sequence σ continues with some infinite reduction σ' from $v[a']\vec{b'}$, including infinitely many safe reductions. From $u[\vec{d_i}, x] \mapsto^* v[x]$ and $\vec{b} \mapsto^* \vec{b'}$ we deduce that

$$t_{i+1}[\vec{d_i}, a]\vec{b} \quad = \quad u[\vec{d_i}, a]\vec{b} \quad \mapsto^* \quad v[a]\vec{b} \quad \mapsto^* \quad v[a]\vec{b'} \quad (1)$$

We proved that there is a reduction σ' including infinitely many safe reductions from $v[a']\vec{b'}$. From $a \mapsto^* a'$ and Lemma 8.9 we deduce that there is a reduction including infinitely many safe reductions from $v[a]\vec{b'}$. From (1) above we conclude that the same holds for $t_{i+1}[\vec{d_i}, a]\vec{b}$.

8. The case of `cond`-rule, namely a sequent $\mathtt{Lbl}(\Pi, l_i) = \Gamma \vdash t_i : N, \vec{A} \to U$ having $t_i[\vec{x}] = \mathtt{cond}(f[\vec{x}], g[\vec{x}])$, and obtained from $\Gamma \vdash f[\vec{x}] : \vec{A} \to U$ and $\Gamma \vdash g[\vec{x}] : N, \vec{A} \to U$. By the induction hypothesis, there is some infinite reduction sequence σ from $t_i[\vec{d_i}]a\vec{b} = \mathtt{cond}(f[\vec{d_i}], g[\vec{d_i}])a\vec{b} : U$ with infinitely many safe reductions, where $\vec{a_i} = a, \vec{b}$. We argue by case.

   (1) *Suppose the reduction sequence σ never contracts the leftmost* `cond`. a and $\vec{b}$ are total, only finitely many reduction on them are possible. Then infinitely many safe reductions take place on $f[\vec{d_i}]$ or $g[\vec{d_i}]$. All these safe reduction are in $f[\vec{d_i}]$, because a reduction inside $g[\vec{d_i}]$ is in the right-hand-side of some `cond`, therefore it is not safe. The safe-infinite reduction from $f[\vec{d_i}]$ can be raised to an infinite reduction from $f[\vec{d_i}](\vec{b})$, with the same safe reductions, therefore with infinite safe reduction. In this case we take $f[\vec{x}], \vec{d_i}, \vec{b}$ as next step of our path in Π: we choose $l_{i+1} = l_i\star(1)$, $\vec{d_{i+1}} = \vec{d_i}$, and $\vec{a_{i+1}} = \vec{b}$. This is an assignment with total terms.
   We have trace-compatibility because: each argument of t_i is connected to some equal argument of $t_{i+1}[\vec{x}] = f[\vec{x}]$, the first argument of $t_i[\vec{x}]$ disappears but it is connected to no N-argument in $f[\vec{x}]$.

   (2) *Suppose the reduction sequence contracts the leftmost* `cond`*-redex at some step.* Then $t_i[\vec{d_i}]a\vec{b} \mapsto^* \mathtt{cond}(f', g')a''\vec{b'}$, with $a'' = 0$ or $a'' = \mathtt{S}(a')$. Then in the first subcase $\mathtt{cond}(f', g')a''\vec{b'} \mapsto f'\vec{b'}$, in the second subcase $\mathtt{cond}(f', g')a''\vec{b'} \mapsto g'a'\vec{b'}$, with $f[\vec{d_i}] \mapsto^* f'$, $g[\vec{d_i}] \mapsto^* g'$, and $\vec{b} \mapsto^* \vec{b'}$. After this `cond`-reduction we have a reduction sequence with infinitely many safe reductions. We prove our thesis by the subcases on a''.

      (i) *Suppose* $a'' = 0$. In this case we take $f[\vec{x}], \vec{d_i}, \vec{b}$ as the next step of our path in Π: we choose $l_{i+1} = l_i\star(1)$, $\vec{d_{i+1}} = \vec{d_i}$, and $\vec{a_{i+1}} = \vec{b}$. This is an assignment with total terms. $f[\vec{d_{i+1}}](\vec{b})$ reduces to $f'\vec{b'}$, then we have infinitely many safe reductions.
      We have trace-compatibility because: each argument of t_i is connected to some equal argument of $t_{i+1}[\vec{x}] = f[\vec{x}]$, the first argument of $t_i[\vec{x}]$ disappears but it is connected to no N-argument in $f[\vec{x}]$.

      (ii) *Suppose* $a'' = \mathtt{S}(a')$. We choose $l_{i+1} = l_i\star(2)$, $\vec{d_{i+1}} = \vec{d_i}$, and $\vec{a_{i+1}} = a', \vec{b}$. This is an assignment with total terms: a' is total because all reduction sequences from a' can be raised from a reduction sequences from $a'' = \mathtt{S}(a')$ with the same safe-reduction. From $a \mapsto^* a''$ we deduce that there are finitely many safe-reductions from a'', therefore finitely many those from a'.
      We have trace-compatibility because: each argument of $t_i[\vec{x}]$ is connected to some equal argument of $t_{i+1}[\vec{x}] = g[\vec{x}]$, but for the first unnamed argument a of $t_i[\vec{x}]$ which is connected the first unnamed argument of $g[\vec{x}]$. This is fine because in the second premise of a `cond` the trace progresses and we have $a' \prec a$, because $a \mapsto^* a'' = \mathtt{S}(a')$.

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

8.1 Strong Normalization for safe reductions in CT-λ and Failure of the Closure under Limit for CT-λ

From this theorem we derive the strong normalization result for safe reductions on terms of `GTC` and for all terms which satisfy the global trace condition.

Corollary 8.11 *Assume $t \in \texttt{CT-}\lambda$ and $n \in N$.*

1. *t is total and all reduction sequences from t include only finitely many 0-safe reductions.*
2. *t is total and all reduction sequences from t include only finitely many n-safe reductions.*
3. *t strongly n-safe-normalizes, and in all infinite reductions from t from some point on all reductions are not n-safe.*
4. *All infinite reduction sequences in $\texttt{CT-}\lambda$ from t have some limit in $\mathbb{\Lambda}$, a limit which is sometimes outside $\texttt{CT-}\lambda$.*

Note that $\texttt{CT-}\lambda$ is *not* **closed under infinite reductions**: the limit of any infinite reduction of any term of $\texttt{CT-}\lambda$ exists and it is unique in $\mathbb{\Lambda}$, yet it is often not in $\texttt{CT-}\lambda$.

As an example, consider a term $f = \lambda x^N.\texttt{cond}(x, f(\texttt{S}x))$, which actually implements the addition. f has a type $N, N \to N$. f is regular, and it is in $\texttt{GTC}$ because the unique infinite branch crosses the right-hand-side of a $\texttt{cond}$ infinitely many times. Hence $f \in \texttt{CT-}\lambda$. From $f(x) \mapsto \texttt{cond}(n, f(\texttt{S}n))$ we deduce that there is a infinite reduction sequence

$$f(x) \mapsto \texttt{cond}(x, f(\texttt{S}(x))) \mapsto \texttt{cond}(x, \texttt{cond}(\texttt{S}(x), f(\texttt{S}^2(x)))) \mapsto \cdots .$$

Hence its limit is $\texttt{cond}(x, \texttt{cond}(\texttt{S}(x), \texttt{cond}(\texttt{S}^2(x), \cdots)))$, which is *not* regular, since it has infinitely many different subterms $0, 1, 2, \ldots$. Therefore it is not in $\texttt{CT-}\lambda$, even though each term finitely reachable from $f(x)$ is in $\texttt{CT-}\lambda$, and the interpretation of f is just the unary algorithm for the addition, because

$$f(\texttt{S}^n(0))(\texttt{S}^m(0)) \mapsto \texttt{cond}(n, \texttt{cond}(\texttt{S}(n), \texttt{cond}(\texttt{S}^2(n), \cdots)))(\texttt{S}^m(0)) \mapsto \texttt{S}^{n+m})(0)$$

This is an unfortunate but common situation for the limit of a sequence in a topology: finiteness (which in our case is expressed by *regulary*) is easily lost under limit. An example: suppose T_n is the complete binary tree of height n and T_∞ is the complete infinite binary tree. Then $\lim_{n\to\infty} T_n = T_\infty$ and all T_n are finite, yet T_∞ is not finite.

8.2 Closed Safe-Normal terms of $\texttt{CT-}\lambda$ of type N are Numeral

This property is not straightforward. There are closed normal terms of N in $\mathbb{\Lambda}$, even regular terms, which are not numerals.

An example is $t = \texttt{cond}(0, \lambda x^N.1)(t)$ ([1], Remark 44, page 33). t has a unique type N. t is normal, because t is not 0 neither is a successor, therefore t cannot be reduced. However t is not a numeral.

Remark that t is regular: the subterms of t are t itself and $\texttt{cond}(0, \lambda x^N.1)$, 0, $\lambda x^N.1$, 1. However, t is not in $\texttt{GTC}$. Indeed, the rightmost path of t is an infinite path $t, t, t, t, \ldots$, never crossing a $\texttt{cond}$, and therefore never progressing.

We check that instead safe-normalization on closed terms of $\texttt{GTC}$ of type N produce a numeral. As an intermediate step in this proof we need the notion of "sound" term.

Definition 8.12 (Sound terms) *A term $t \in \texttt{CT-}\lambda$ is sound if at least one of the following conditions holds.*

1. *t is open*
2. *t has some 0-safe redex*
3. *$t = \lambda x.u$ or $t = \texttt{cond}(f, g)$ for some u, f, g and t has an arrow type*
4. *t is a numeral*

Proposition 8.13 (Closed Safe-Normal terms of Type N) *1. All $t \in \texttt{GTC}$ are sound*
2. If $t \in \texttt{GTC}$, t is closed and 0-safe-normal and $t : N$ then $t \in \texttt{Num}$ (t is a numeral)

Proof 1. We assume that $t \in \texttt{GTC}$ in order to prove that t is sound. We argue by induction on the size of the 0-safe level of t: this size is finite because $t \in \texttt{GTC}$. We consider one case for each possible first symbol of t.

(1) $t = x$. Then t is open.

(2) $t = 0$. Then t is a numeral.

(3) $t = \mathtt{S}(u)$. Then $u : N$ and u has a smaller 0-safe level, therefore u is sound. If u is open then t is open, if u has a 0-safe redex then so does t, u cannot have an arrow type, and if u is a numeral then t is a numeral.

(4) $t = u(v)$. Then $u : A \to B$, $u : A$ and u, v have a smaller 0-safe level, therefore u, v are sound. If u or v are open then t is open.
If u or v have a 0-safe redex then so does t.
Suppose $u = \lambda x.v, \mathtt{cond}(f, g)$. If $u = \lambda x.v$ then $t = u(v)$ is a 0-safe β-redex. If $u = \mathtt{cond}(f, g)$ then $A = N$ and $v : N$. v is not open nor has a a 0-safe redex, and v has no arrow type. The only possibility left is that v is a numeral: we conclude that $t = \mathtt{cond}(f, g)(v)$ is a $\mathtt{cond}$-redex, therefore t is sound.
The only possibility left for u is that it is a numeral. But this cannot be, because u has an arrow type.

(5) $t = \lambda x.u$. Then t is sound.

(6) $t = \mathtt{cond}(f, g)$. Then t is sound.

2. Assume that $t \in \mathtt{GTC}$, t is closed and 0-safe-normal and $t : N$. Then t is sound by the previous point and the only possibility left is that t is a numeral.

9 Infinite Church-Rosser for Infinite Lambda Terms

We already proved (8.11) that all terms of CT-λ have some safe normal form, some n-safe normal form for all $n \in N$, and some full normal form in the limit. In this section prove the corresponding uniqueness results, namely:

1. **"Uniqueness of the safe normal form"**. For all $t, u, v \in$ CT-λ, if $t \mapsto u$ and $t \mapsto v$ and u, v are safe-normal then $u \approx_{0,\mathtt{cond}} v$. That is: u, v are equal outside the right-hand side of all cond-sub-terms.
2. **"Uniqueness of the n-safe normal form"**. For all $t, u, v \in$ CT-λ, if $t \mapsto u$ and $t \mapsto v$ and u, v are n-safe-normal then $u \approx_{n,\mathtt{cond}} v$. That is: u, v are equal outside the right-hand side of all cond-sub-terms of cond-depth equal to n.
3. **"Uniqueness of the limit normal form"**. For all $t, u, v \in$ CT-λ, if u, v are limit normal form of t then $u = v$.

The last point implies that if $t, u, v \in$ CT-λ, if $t \mapsto u$ and $t \mapsto v$, then the limit normal forms of u, v, being limit normal form of t, are the same. This means that we are looking for some kind Church-Rosser property, at least in the limit.

Our first idea (which turns out to be wrong) is to prove a full Church-Rosser property for CT-λ: for all $t, u, v \in$ CT-λ, if $t \mapsto u$ and $t \mapsto v$ then for some $w \in$ CT-λ we have $u \mapsto w$ and $v \mapsto w$. This property is false: for some $t \in$ CT-λ, finding a common reduction of u, v takes infinitely many steps. The reason is that if we have infinitely many occurrences of a variable x, then a reduction $(\lambda x.t)(u) \mapsto t[u/x]$ can multiply infinitely many times the redexes in u. Here is an example in which this happens.

Example 9.1 (A term $t \in$ CT-λ for which Church-Rosser fails) *Let $b = \mathtt{cond}(x^N, b) : N \to N$ a normal form and $t = (\lambda x^N.b)(r) : N \to N$, where $r = (\lambda x^N.x)(3)$ is some β-redex. We have $t \in$ CT-λ. Indeed,*

1. *t is regular by construction.*
2. *We have $t \in$ GTC, because the unique infinite path of t is $t, \lambda x^N.b, b, b, b, \ldots$, and the unique unnamed argument of $b : N \to N$ in the path progresses infinitely many times.*

Now we check that:

Lemma 9.1 *Church-Rosser fails for the term t defined above (Ex. 9.1), hence for* CT-λ

Proof With a β-reduction on t itself we obtain $t \mapsto b[r/x^N]$. With another β reduction $r \mapsto 3$ we obtain $t \mapsto (\lambda x^N.b)(3)$. We expect $b[3/x^N]$ as common normal form for the two β-reductions. But we have $b[r/x^N] = \mathtt{cond}(r, b[r/x^N])$, that is, we have replicated the redex r infinitely many times in $b[r/x^N]$. Therefore to reduce $b[r/x^N]$ to $b[3/x^N]$ takes infinitely many β reductions of the form $r \mapsto 3$, and for *no finite reduction we have* $b[r/x^N] \mapsto b[3/x^N]$.

$b[r/x^N]$ is a term without normal form: all its normal form are in the limit.

We proved that Church-Rosser is false for CT-λ.

As we anticipated, we will recover a weak form of Church-Rosser in the limit for CT-λ.

We say that $t \in$ CT-λ reduces in the limit to $u \in$ CT-λ, and we write $t \mapsto_{\lim} u$, if $t \mapsto^* u$ or $t = \lim_{n\to\infty} t_n$ for some infinite sequence $t = t_0 \mapsto t_1 \mapsto t_2 \mapsto \ldots$. Remark that we include finite reductions as trivial cases of limit reductions. We say that u is a limit normal form of t if and only if $t \mapsto_{\lim} u$ and u is normal.

We say that a a binary relation $\mapsto$ on CT-λ is a *sub-limit extension of* $\mapsto$ if it is between $\mapsto$ and $\mapsto_{\lim}$:

$$\mapsto \quad \subseteq \quad \mapsto \quad \subseteq \quad \mapsto_{\lim}$$

The existence of a confluent sub-limit extension of $\mapsto$ is a sufficient condition in order to conclude our unicity results.

Lemma 9.2 (Sufficient condition for Unicity of normal form) *Assume there is some confluent sub-limit extension $\mapsto$ of $\mapsto$ on terms of* CT-λ. *Then:*

1. *Safe-normal forms and n-safe normal forms for any $n \in N$ are unique up to $\approx_{n,\mathtt{cond}}$.*

2. Limit normal form are unique.

Proof Let $\mapsto^*$ be the reflexive and transitive closure of $\mapsto$: from $\mapsto \subseteq \mapsto_{\lim}$ we deduce that $\mapsto^* \subseteq \mapsto^* \subseteq \mapsto^*_{\lim}$. From the assumption that $\mapsto$ is confluent we deduce that $\mapsto^*$ is confluent.

Now we prove:

1. *Unicity of n-safe normal forms up to* $\approx_{n,\mathtt{cond}}$. Suppose $t \mapsto^* u, v$ and u, v are n-safe normal, in order to prove that $u \approx_{n,\mathtt{cond}} v$. From $\mapsto^* \subseteq \mapsto^*$ we have $t \mapsto^* u, v$. From confluence of $\mapsto^*$ we deduce that for some $w \in \mathtt{CT}\text{-}\lambda$ we have $u, v \mapsto^* w$. From $\mapsto^* \subseteq \mapsto^*_{\lim}$ we deduce that $u, v \mapsto^*_{\lim} w$. From u, v n-safe normal we deduce that all reductions on u, v take place in the rigth-hand-side of some depth-n $\mathtt{cond}$. Thus, for all $z \in \mathtt{CT}\text{-}\lambda$ we have $u \mapsto z$ implies $u \approx_{n,\mathtt{cond}} z$.
   (1) By induction on the reduction length we deduce that $u \mapsto^* z$ implies $u \approx_{n,\mathtt{cond}} z$.
   (2) By taking the limit we obtain that $u \mapsto_{\lim} z$ implies $u \approx_{n,\mathtt{cond}} z$.
   (3) By induction on the reduction length again we get that $u \mapsto^*_{\lim} z$ implies $u \approx_{n,\mathtt{cond}} z$.

   By switching the role of u and v we deduce that $v \mapsto^*_{\lim} z$ implies $v \approx_{n,\mathtt{cond}} z$. Eventually, from $u, v \mapsto^*_{\lim} w$ we conclude that $u \approx_{n,\mathtt{cond}} w \approx_{n,\mathtt{cond}} v$, as we wished to show.
2. *Unicity of limit normal forms.* Suppose $t \mapsto_{\lim} u, v$ and u, v are normal, in order to prove that $u = v$. Let $n \in N$. From the finiteness of n-safe reduction we deduce that $t \mapsto^* u' \mapsto_{\lim} u$ for some $u' \in \mathtt{CT}\text{-}\lambda$ such that there are no n-safe reductions from u' to u, that is: $u' \approx_{n,\mathtt{cond}} u$. From u n-safe normal we deduce that u' is *n-safe normal*. By switching the role of u and v we deduce that $t \mapsto^* v' \mapsto_{\lim} v$ for some *n-safe normal* $v' \in \mathtt{CT}\text{-}\lambda$ such that $v' \approx_{n,\mathtt{cond}} v$.
   From unicity of n-safe normal form we deduce that $u' \approx_{n,\mathtt{cond}} v'$. From $u' \approx_{n,\mathtt{cond}} u$ and $v' \approx_{n,\mathtt{cond}} v$ we obtain that $u \approx_{n,\mathtt{cond}} v$, for all $n \in N$. By definition of equality on infinite terms we conclude that $u = v$, as we wished to show.

In order to conclude unicity we have to define some sub-limit extension $\mapsto$ of $\mapsto$. $\mapsto$ will select any subset of the redexes of any $t \in \mathtt{CT}\text{-}\lambda$ and contracts all of them at the same time. This operation can produce new redexes, but the new redexes will not be contracted. We will need the global trace condition in order to prove that $\mapsto$ is well-defined.

There is a Church-Rosser proof in Barendregt's book (page 61, Lemma 3.2.6), in which a parallel reduction is defined on induction on the term. However, it not obvious how to adapt this definition of parallel reductions to infinite terms, even for infinite terms with the global trace condition, because we lack induction.

Instead, we will define a parallel reduction $\mapsto$ following the second Church-Rosser proof in the same book (page 282, Theorem 10.1.11). The idea is labeling the main constructor of a redex in λ-calculus, in order to distinguish which redexes we contract in a given reduction step, and which redexes we do not contract.

We first define a *"labelled circular λ-calculus"* $\mathtt{lab}\text{-}\mathtt{CT}\text{-}\lambda$ as a version of $\mathtt{CT}\text{-}\lambda$ with applications labeled by some $i \in N$: $\mathtt{ap}_0, \mathtt{ap}_1, \mathtt{ap}_2, \ldots$. We assume the same reduction rules for all $\mathtt{ap}_i$, $i \in N$:

$$\mathtt{ap}_i(\lambda x.t, u) \mapsto t[u/x], \quad \mathtt{ap}_i(\mathtt{cond}(f,g)(0)) \mapsto f, \quad \mathtt{ap}_i(\mathtt{cond}(f,g)(S(t))) \mapsto \mathtt{ap}_0(f,t)$$

When $i \neq 0$, we use the subscript i as a label to trace what happens to a subterm $\mathtt{ap}_i(t,u)$ during a reduction. When we reduce a redex $\mathtt{ap}_i(\ldots)$, the label i on the first $\mathtt{ap}$-symbol of a redex disappears. In the reduction $\mathtt{ap}_i(\mathtt{cond}(f,g)(S(t))) \mapsto \mathtt{ap}_0(f,t)$ there is new application born after reduction. We assign to it the label 0: this new application has a trivial labelling.

When $i = 0$ we identify $\mathtt{ap}_0$ with $\mathtt{ap}$: in this way $\mathtt{CT}\text{-}\lambda$ is a proper subset of $\mathtt{lab}\text{-}\mathtt{CT}\text{-}\lambda$. We define a projection $\overline{(\cdot)} : \mathtt{lab}\text{-}\mathtt{CT}\text{-}\lambda \to \mathtt{CT}\text{-}\lambda$ by hereditarily replacing all subterms $\mathtt{ap}_i(u,v)$ of $\mathtt{lab}\text{-}\mathtt{CT}\text{-}\lambda$ with $\mathtt{ap}_0(u,v)$. For all $t \in \mathtt{lab}\text{-}\mathtt{CT}\text{-}\lambda$ we have $\overline{t} = t$ if and only if $t \in \mathtt{CT}\text{-}\lambda$.

We define a relation between a term and a set of labels.

Definition 9.3 (X-applications, X-redexes and X-terms) *Assume that $t \in\in \mathtt{CT}\text{-}\lambda$, $r = \mathtt{ap}_i(\ldots) \in \mathtt{CT}\text{-}\lambda$ and $X \subseteq N$, $i \in N$.*

1. r is an i-application, and an i-redex if r is a redex.

2. *r is an X-application (an X-redex) if and only if r is an i-application (an i-redex) for some $i \in X$.*
3. *t is an X-term if and only if all redexes in t are X-redexes.*

For all $t \in \texttt{lab-CT-}\lambda$, all sets $X \subseteq N$ of labels we define a map $\phi_X(t)$ reducing all i-redexes in t for all $i \in X$. $\phi_X(t)$ is the limit of a family $\phi_{n,X}(t)$ for $n \to \infty$, a family which is defined by induction on n.

Definition 9.4 (The maps $\phi_{n,X}$ and ϕ_X) *Assume $t \in \texttt{lab-CT-}\lambda$. We set $\phi_{0,X}(t) = t$. If $n > 0$ then we define ϕ_n out of ϕ_{n-1}:*

1. *$\phi_{n,X}(x^T) = x^T$ and $\phi_{n,X}(0) = 0$.*
2. *$\phi_{n,X}(\mathtt{S}(t)) = \mathtt{S}(\phi_{n-1,X}(t))$.*
3. *Let $r = \mathtt{ap}_i(t,u)$.*
   (1) Suppose $i \notin X$ or r is not a redex. Then we set
   $$\phi_{n,X}(r) = \mathtt{ap}_i(\phi_{n-1,X}(t), \phi_{n-1,X}(u))$$
   (2) Suppose $i \in X$ and r is a redex whose contraction is s. Then we set
   $$\phi_{n,X}(r) = \phi_{n-1,X}(s)$$
4. *$\phi_{n,X}(\lambda x.z) = \lambda x.\phi_{n-1,X}(z)$.*
5. *$\phi_{n,X}(\mathtt{cond}(f,g)) = \mathtt{cond}(\phi_{n-1,X}(f), \phi_{n-1,X}(g))$.*

We set $\phi_X(t) = \lim_{n\to\infty}(\phi_{n,X})(t)$.

By induction on n we immediately prove that $\phi_{n,X}$ is a well defined map $\texttt{lab-CT-}\lambda \to \texttt{lab-CT-}\lambda$. Instead, it is not self-evident that $\phi_X(t)$ is a total tree: this requires co-induction, and in the case $0 \in X$ even Finiteness of safe reductions (8.11).

Again to prove that $\phi_X(t)$ is a total tree, we also the notion of X-chain and X-length for labelled terms.

Definition 9.5 (X-chain and X-length) *Let $t \in \texttt{lab-CT-}\lambda$.*

1. *An X-chain from t is any finite sequence $t_0, \ldots, t_k$ such that: $t = t_0$, for all $a < k$ we have t_a some i-redex for some $i \in X$, and t_{a+1} is the contractum of t_a.*
2. *The X-length is the lenght of the longest X-chain, if finite. If the X-length exists we write $\mathtt{len}_X(t) = k$. In this case call t_k the X-form of t and we write $\mathtt{form}_X(t) = t_k$.*

The longest X-path $t_0, \ldots, t_k, \ldots$ is unique because the contractum of any redex is unique. Any X-path is a path of safe reductions. By Finiteness of safe reductions (8.11) the longest X-path is *always finite*. By definition, the X-form t_k of t is no X-redex.

The last ingredient is the notion of head of a term of $\texttt{lab-CT-}\lambda$.

Definition 9.6 (Head) *The head* $\mathrm{head}(t)$ *of a term $\lambda x^A.t$ are the first two symbols of the term, namely* $\mathrm{head}(t) = \lambda, x^A$ *(with apices). The head of any other term: $t = x^T, 0, \mathtt{S}(u), \mathtt{ap}_i(v,w), \mathtt{cond}(f,g)$ is its first symbol (with its indexes and apices) respectively:* $\mathrm{head}(t) = x^T, 0, \mathtt{S}, \mathtt{ap}_i, \mathtt{cond}$.

Eventually we can prove:

Lemma 9.7 ($\phi_X(t)$ is a total tree) *For all $t \in \texttt{lab-CT-}\lambda$, $\phi_X(t)$ is a total tree.*

Proof Let $k \in N$ be the X-length of t. Then the longest X-chain is some $t_0, \ldots, t_k$, with t_k no X-redex. By definition of $\phi_{n,X}$ we deduce that for all $n \geq k$ we have $\phi_{n,X}(t) = phi_{n-1,X}(t_1) = phi_{n-2,X}(t_2) = \ldots = phi_{n-k,X}(t_k)$. We assumed that t_k is no i-redex, for any $i \in X$. By definition of $phi_{n-k,X}(t_k)$, we deduce that for all $n \geq k+1$, if we abbreviate $\psi \equiv \phi_{n-(k+1),X}$, then we have that $\phi_{n-k,X}(t_k)$ is one among
$$x, \quad 0, \quad \mathtt{S}(\psi(u)), \quad \mathtt{ap}_i(\psi(v),\psi(w)), \quad \lambda x.\psi(z), \quad \mathtt{cond}(\psi(f),\psi(g))$$
according if t is
$$x, \quad 0, \quad \mathtt{S}(u), \quad \mathtt{ap}_i(v,w), \quad \lambda x.z, \quad \mathtt{cond}(f,g)$$
In all cases we proved that the node $\phi_{n,X}(t)$ has the same head ξ for all $n \geq k+1$, and therefore that $\phi_X(t) = \lim_{n\to\infty}(\phi_{n,X}) = \xi$ exists.

We define an equality $t =_n u$, which means: the subterms of t, u of distance $< n$ from the root are the same.

Definition 9.8 (n-equality on Terms of CT-λ) *Let $a, a', u, u', v, v', z, z', f, f', g, g' \in$ CT-λ. The predicate $=_n$ is the smallest predicate closed w.r.t. the following operations.*

1. *We set $a =_0 a'$ for all $a, a' \in$ CT-λ.*
2. *If $n > 0$ and*

$$u \approx_{n-1} u' \quad v \approx_{n-1} v' \quad w \approx_{n-1} w' \quad z \approx_{n-1} z' \quad f \approx_{n-1} f' \quad g \approx_{n-1} g'$$

then:

$$x \approx_n x, \quad 0 \approx_n 0, \quad \mathtt{S}(u) \approx_n \mathtt{S}(u'), \quad \mathtt{ap}_i(u, v) \approx_n \mathtt{ap}_i(u', v') \quad \lambda x.z \approx_n \lambda x.z', \quad \mathtt{cond}(f, g) \approx_n \mathtt{cond}(f', g')$$

Then $t = u$ if and only if $t =_n u$ for all $n \in N$. Both $\phi_{n,X}$ and ϕ_X are continuous maps, w.r.t. the topology on lab-CT-λ whose basic opens are all sets $O_{n,t} = \{u \in \text{lab-CT-}\lambda | u \approx_n t\}$ for $t \in$ lab-CT-λ. Usually, the basic open $O_{n,t}$ is much smaller than the n-safe-level of t.

For any list $l \in \{1, 2\}^*$ of elements $1, 2$, if $l \in |t|$ is the address of some subterm u of t then we set $t_l = u$ otherwise we set $t_l = \perp$.

We sum up in a Lemma how to prove an equation $\alpha(t) = \beta(t)$ for $t \in$ lab-CT-λ by proving the existence of an identical map (in fact, an identical bisimulation) between $\alpha(t)$ and $\beta(t)$.

Lemma 9.9 (Identity between terms of lab-CT-λ) *Assume $\alpha, \beta :$ lab-CT-$\lambda \to$ lab-CT-λ. Suppose that:*

1. $\mathrm{head}(\alpha(t)) = \mathrm{head}(\beta(t))$
2. *Either $\alpha(t) = \beta(t)$, or for all $i \leq a =$ the arity of the tree $\alpha(t)$, there is some $u \in$ lab-CT-λ such that the i-th sub-terms of $\alpha(t)$ and of $\beta(t)$ are, respectively, $\alpha(u)$ and $\beta(u)$.*

Then $\alpha(t) = \beta(t)$ for all $t \in$ lab-CT-λ.

Assume that if t is an $X \cup Y$-term, that is, that all i-applications of t are redexes for some $i \in X \cup Y$ (Def. 9.3). We claim that if we apply ϕ_X and ϕ_Y consecutively, that is, if we remove first all redexes with label in X then all redexes with label in in Y, as a result we remove all redexes with label in $X \cup Y$.

The first step in this proof is to characterize the value of $\phi_X(t)$.

Lemma 9.10 (Characterizing $\phi_X(t)$) *Assume that $t \in$ lab-CT-λ and $X \subseteq N$.*

1. *If t is no X-redex and t is*

$$x, \quad 0, \quad \mathtt{S}(u), \quad \mathtt{ap}_i(v, w), \quad \lambda x.z, \quad \mathtt{cond}(f, g)$$

then $\phi_X(t)$ is, respectively:

$$x, \quad 0, \quad \mathtt{S}(\phi_X(u)), \quad \mathtt{ap}_i(\phi_X(v), \phi_X(w)), \quad \lambda x.\phi_X(z), \quad \mathtt{cond}(\phi_X(f), \phi_X(g))$$

2. *If t' is the X-form of t then $\phi_X(t) = \phi_X(t')$.*

Proof 1. Assume that $n \in N$, that t is no i-redex for any $i \in X$ and t is

$$x, \quad 0, \quad \mathtt{S}(u), \quad \mathtt{ap}_i(v, w), \quad \lambda x.z, \quad \mathtt{cond}(f, g)$$

Then $\phi_{n+1,X}(t)$ is, respectively:

$$x, \quad 0, \quad \mathtt{S}(\phi_{n,X}(u)), \quad \mathtt{ap}_i(\phi_{n,X}(v), \phi_{n,X}(w)), \quad \lambda x.\phi_{n,X}(z), \quad \mathtt{cond}(\phi_{n,X}(f), \phi_{n,X}(g))$$

We deduce that $\phi_X(t)$, being the limit of $\phi_{n+1,X}(t)$, is equal to

$$x, \quad 0, \quad \mathtt{S}(\phi_X(u)), \quad \mathtt{ap}_i(\phi_X(v), \phi_X(w)), \quad \lambda x.\phi_X(z), \quad \mathtt{cond}(\phi_X(f), \phi_X(g))$$

2. Assume that $t = t_0 \to t_1 \to \ldots \to t_k = t'$ is the longest X-reduction path from t, hence that t' is the X-form of t. We have to prove that $\phi_X(t) = \phi_X(t')$. For all $n \in N$ by definition of ϕ_X we have

$$\phi_{n+k,X}(t) = \phi_{n+k,X}(t_0) = \phi_{n+k-1,X}(t_1) = \ldots = \phi_{n+k-k,X}(t_k) = \phi_{n,X}(t')$$

By taking the limit for $n \to \infty$ we conclude that $\phi_X(t) = \phi_X(t')$, as wished.

The next step is to prove that X-terms are closed under substitution.

Lemma 9.11 *Assume that $t \in$ lab-CT-λ and $X \subseteq N$. If t, u are X-terms (all X-applications in t, u are redexes), then $t[u/x]$ is an X-term.*

Proof Assume that all X-applications in t, u are redexes in order to prove that all X-applications in $t[u/x]$ are redexes. If $l \in \{1,2\}^*$ is the address of a subterm of $t[u/x]$ we have two possibilities:

1. either $l = l_1 \star l_2$ with $l_1 \in |t|$, $l_2 \in |u|$ and $t[u/x]_l = u_{l_2}$,
2. or $l = l_1 \in |t|$ and $t[u/x]_l = t_{l_1}[u/x]$.

Assume $t[u/x]_l$ is an i-application in order to prove that $t[u/x]_l$ is a redex.

1. In the first case, we have that $t[u/x]_l = u_{l_2}$ is an i-application. By our assumption on u, we deduce than u_{l_2} is a redex.
2. In the second case we have $t[u/x]_l = t_{l_1}[u/x]$. Either $t_{l_1} = x$ or not. If $t_{l_1} = x$ then $t[u/x]_l = u_{\mathtt{nil}}$ and we are reduced to the previous case. Suppose $t_{l_1} \neq x$. Then the head of t_{l_1} and $t[u/x]_l$ are the same. From $t[u/x]_l$ we deduce that t_{l_1} is an i-application, then by assumption of t that t_{l_1} is redex. Redexes are closed under substitution, therefore $t_{l_1}[u/x] = t[u/x]_l$ is a redex. We conclude that $t[u/x]_l = t_{l_1}[u/x]$ is a redex.

We can now prove our thesis $\phi_X(\phi_Y(t)) = \phi_{X\cup Y}(t)$, from an auxiliary equation $\phi_X(t)[\phi_X(u)/x] = \phi_X(t[u/x])$.

Lemma 9.12 (Composition and Union) *Assume that $t \in$ lab-CT-λ and $X \subseteq N$.*

1. *Commutation with substitution. If t is an X-term, then $\phi_X(t)[\phi_X(u)/x] = \phi_X(t[u/x])$.*
2. *Composition and Union. If $0 \notin X \cup Y$ and t is an X-term (all $X \cup Y$-applications in t are redexes), then $\phi_X(\phi_Y(t)) = \phi_{X\cup Y}(t)$.*

Proof 1. *Commutation with substitution.* Assume that all X-applications in t are redexes, in order to prove that $\phi_X(t)[\phi_X(a)/x] = \phi_X(t[a/x])$. We abbreviate $\sigma = [\phi_X(a)/x]$ and $\tau = [a/x]$: then we have to prove $\phi_X(t)\sigma = \phi_X(t\tau)$.
By Lemma 9.9, applied to t, we have to prove that $\mathrm{head}(\phi_X(t)\sigma) = \mathrm{head}(\phi_X(t\tau))$ and that either $\phi_X(t)\sigma = \phi_X(t\tau)$, or any two immediate subterms in the same position in $\phi_X(t)\sigma$ and $\phi_X(t\tau)$ are of the form $\phi_X(t')\sigma$ and $\phi_X(t'\tau)$ for some t'. We argue by cases on the condition: *"t X-redex"*.

(1) *First sub-case.* We first assume that t is *no* X-redex. Then t is

$$x, \quad y \text{ (with } y \neq x), \quad 0, \quad \mathtt{S}(u), \quad \mathtt{ap}_i(v,w), \quad \lambda x.z, \quad \mathtt{cond}(f,g)$$

for some X-terms $u, v, w, z, f, g \in$ lab-CT-λ. In the case $t = \mathtt{ap}_i(v,w)$, from t not X-redex we deduce that t is no X-application, that is, that $i \notin X$. By definition, $x\sigma = \phi_X(a)$ and $y\sigma = y$ for all $y \not\equiv x$. We deduce that $\phi_X(t)\sigma$ is

$$x\sigma = \phi_X(a), \quad y\sigma = y, \quad 0, \quad \mathtt{S}(\phi_X(u)\sigma), \quad \mathtt{ap}_i(\phi_X(v)\sigma, \phi_X(w)\sigma), \quad \lambda x.\phi_X(z)\sigma, \quad \mathtt{cond}(\phi_X(f)\sigma, \phi_X(g)\sigma)$$

By definition, $x\tau = a$ and $y\sigma = y$ for all $y \not\equiv x$. We deduce that $\phi_X(t\tau)$ is, according to the shape of t:

$$\phi_X(x\tau) = \phi_X(a), \quad \phi_X(y\tau) = \phi_X(y) = y, \quad 0, \quad \mathtt{S}(\phi_X(u\tau)), \quad \mathtt{ap}_i(\phi_X(v\tau), \phi_X(w\tau)), \quad \lambda x.\phi_X(z\tau), \quad \mathtt{cond}(\phi_X(f\tau), \phi_X(g\tau))$$

We conclude that any two immediate subterms in the same position in $\phi_X(t)\sigma$ and $\phi_X(t\tau)$ are of the form $\phi_X(t')\sigma$ and $\phi_X(t'\tau)$ for some t', as we wished to show.

(2) *Second sub-case.* We assume that t is *some* X-*redex* with X-form t', with some maximal X-reduction $t \to^* t'$: all terms in the path but t' are X-redexes, while t' is no X-redex. Substitution preserves X-redexes, therefore if we apply τ we obtain some (possibly not maximal) X-reduction $t\tau \to^* t'\tau$, all terms in the path but $t'\sigma$ are X-redexes, while $t'\sigma$ is *could* be an X-redex or not.
By definition of ϕ we have $\phi_X(t)\sigma = \phi_X(t')\sigma$ and $\phi_X(t\tau) = \phi_X(t'\tau)$. From t' no X-redex and the sub-case 1.1 above, any two immediate subterms in the same position of $\phi_X(t'\tau)$ and $\phi_X(t'\sigma)$ are of the form $\phi_X(t''\tau)$ and $\phi_X(t''\sigma)$ for some t'', as we wished to show.

2. Assume that $0 \notin X \cup Y$ and all $X \cup Y$-applications in t are redexes, in order to prove that $\phi_X(\phi_Y(t)) = \phi_{X\cup Y}(t)$.
By Lemma 9.9, applied to t, we have to prove that either $\phi_X(\phi_Y(t)) = \phi_{X\cup Y}(t)$, or $\mathrm{head}(\phi_X(\phi_Y(t))) = \mathrm{head}(\phi_{X\cup Y}(t))$ and that any two immediate subterms in the same position of the two terms are of the form $\phi_X(\phi_Y(t'))$ and $\phi_{X\cup Y}(t')$, for some X-term $t' \in \mathtt{CT}\text{-}\lambda$. We argue by cases on the condition *"t X-redex"*.

(1) *First Sub-case.* We first assume that t is *no* X-redex. Then t is

$$x, 0, \quad \mathtt{S}(u), \quad \mathtt{ap}_i(v,w), \quad \lambda x.z, \quad \mathtt{cond}(f,g)$$

for some X-terms $u, v, w, z, f, g \in \mathtt{lab}\text{-}\mathtt{CT}\text{-}\lambda$. In the case $t = \mathtt{ap}_i(v,w)$, from t not $X \cup Y$-redex we deduce that t is no $X \cup Y$-application, that is, **that** $i \notin X \cup Y$. Then $\phi_Y(t)$ is

$$x, \quad 0, \quad \mathtt{S}(\phi_Y(u)), \quad \mathtt{ap}_i(\phi_Y(v), \phi_Y(w)), \quad \lambda x.\phi_Y(z), \quad \mathtt{cond}(\phi_Y(f), \phi_Y(g))$$

Again because $i \notin X \cup Y$, we have that $\phi_X(\phi_Y(t))$ is, according to the shape of t:

$$x, \quad 0, \quad \mathtt{S}(\phi_X(\phi_Y(u))), \quad \mathtt{ap}_i(\phi_X(\phi_Y(v)), \phi_X(\phi_Y(w))), \quad \lambda x.\phi_X(\phi_Y(z)), \quad \mathtt{cond}(\phi_X(\phi_Y(f)), \phi_X(\phi_Y(g)))$$

Again because $i \notin X \cup Y$, we have that $\phi_{X\cup Y}(t)$ is, according to the shape of t:

$$x, \quad 0, \quad \mathtt{S}(\phi_{X\cup Y}(u)), \quad \mathtt{ap}_i(\phi_{X\cup Y}(v), \phi_{X\cup Y}(w)), \quad \lambda x.\phi_{X\cup Y}(z), \quad \mathtt{cond}(\phi_{X\cup Y}(f), \phi_{X\cup Y}(g))$$

We conclude that any two immediate subterms in the same position of $\phi_X(\phi_Y(t))$ and $\phi_{X\cup Y}(t)$ are of the form $\phi_X(\phi_Y(t'))$ and $\phi_{X\cup Y}(t')$, for some X-term $t' \in \mathtt{CT}\text{-}\lambda$, as we wished to show.

(2) *Second Sub-case.*
We assume that t is *some* $X \cup Y$-*redex* with $X \cup Y$-form t', with some maximal X-reduction $t = t_0 \to t_1 \ldots \to t_k = t'$, of length $k > 0$. All terms in the X-path but t' are $X \cup Y$-redexes, in particular we have $t = \mathtt{ap}_i(\ldots)$ for some $i \in X \cup Y$, while t' is no $X \cup Y$-redex. We argue by induction on k. By definition of ϕ we have $\phi_{X\cup Y}(t) = \phi_{X\cup Y}(t_1)$. Suppose that $i \in Y$. Then $\phi_X(\phi_Y(t)) = \phi_X(\phi_Y(t_1))$. By induction on k we deduce our thesis: that any two immediate subterms in the same position of $\phi_X(\phi_Y(t_1))$ and $\phi_{X\cup Y}(t_1)$ are of the form $\phi_X(\phi_Y(t'))$ and $\phi_{X\cup Y}(t')$, for some X-term $t' \in \mathtt{CT}\text{-}\lambda$. Suppose that $i \notin Y$. From $i \in X \cup Y$ we deduce that $i \in X$. We argue by sub-sub-cases on the first reduction $t_0 \to t_1$.

Suppose $t = t_0 = \mathtt{ap}_i(\lambda x.u, v) \mapsto u[v/x] = t_1$ is a β-redex, for some X-terms u, v. By definition of ϕ and $i \in X$, $i \notin Y$ we deduce $\phi_X(\phi_Y(t)) =$ (by $i \notin Y$) $\phi_X(\mathtt{ap}_i(\lambda x.\phi_Y(u), \phi_Y(v))) =$ (by $i \in X$) $\phi_X(\phi_Y(u)[\phi_Y(v)/x]) =$ (by commutation of ϕ and substitution, point 1. above) $\phi_X(\phi_Y(u[v/x])) = \phi_{X\cup Y}(t_1)$,
By induction on k we deduce our thesis: that any two immediate subterms in the same position of $\phi_X(\phi_Y(t_1))$ and $\phi_{X\cup Y}(t_1)$ are of the form $\phi_X(\phi_Y(t'))$ and $\phi_{X\cup Y}(t')$, for some X-term $t' \in \mathtt{CT}\text{-}\lambda$.

Suppose $t = t_0 = \mathtt{ap}_i(\mathtt{cond}(f,g), 0) \mapsto f = t_1$ is a $\mathtt{cond}, 0$-redex, for some X-terms f, g. From $i \notin Y$, $i \in X$ and definition of ϕ we deduce $\phi_X(\phi_Y(t)) =$ (by $i \notin Y$) $\phi_X(\mathtt{ap}_i(\mathtt{cond}(\phi_Y(f), \phi_Y(g), 0)$ (by $i \in X$) $\phi_X(\phi_Y(f)) = \phi_X(\phi_Y(t_1))$..

By induction on k we deduce our thesis: that any two immediate subterms in the same position of $\phi_X(\phi_Y(t_1))$ and $\phi_{X\cup Y}(t_1)$ are of the form $\phi_X(\phi_Y(t'))$ and $\phi_{X\cup Y}(t')$, for some X-term $t' \in \texttt{CT-}\lambda$.

Suppose $t = t_0 = \texttt{ap}_i(\texttt{cond}(f,g),\texttt{S}(h)) \mapsto g(h) = t_1$ is a $\texttt{cond},\texttt{S}$-redex, with $i \in X \cup Y$. By $0 \notin X \cup Y$ (one of our assumptions), by $i \notin Y$, $i \in X$ and definition of ϕ we deduce $\phi_X(\phi_Y(t)) = \phi_X(\texttt{ap}_i(\texttt{cond}(\phi_Y(f),\phi_Y(g)),\texttt{S}(\phi_Y(h)) =$ (by $i \in X$) $\phi_X(\texttt{ap}_0(\phi_Y(g),\phi_Y(h))) =$ (by $0 \notin X \cup Y$) $\texttt{ap}_0(\phi_X(\phi_Y(g)),\phi_X(\phi_Y(h)))$, and $\phi_{X\cup Y}(t_1) = \texttt{ap}_0(\phi_{X\cup Y}(g),\phi_{X\cup Y}(h))$
By induction on k we deduce our thesis: that any two immediate subterms in the same position of $\phi_X(\phi_Y(t_1))$ and $\phi_{X\cup Y}(t_1)$ are of the form $\phi_X(\phi_Y(t'))$ and $\phi_{X\cup Y}(t')$, for some X-term $t' \in \texttt{CT-}\lambda$.

We conclude Church-Rosser for set of reductions on $\texttt{lab-CT-}\lambda$.

Lemma 9.13 (Church-Rosser for the reduction set $\phi_X(t)$) *Assume that all i-applications of t are redexes for all $i \in X \cup Y$, and $0 \notin X \cup Y$. Then $\phi_X(\phi_Y(t)) = \phi_Y(\phi_X(t))$, for all $t \in \texttt{lab-CT-}\lambda$.*

Proof Assume $t \in \texttt{lab-CT-}\lambda$. From $Y \cup X = X \cup Y$ we deduce that t is a $Y \cup X$-term and $0 \notin X \cup Y$. We conclude that $\phi_X(\phi_Y(t)) =$ (by Lemma 9.12.2.) $\phi_{X\cup Y}(t) = \phi_{Y\cup X}(t) =$ (by Lemma 9.12.2.) $\phi_Y(\phi_X(t))$, as we wished to show.

We can move the Church-Rosser result to $\texttt{CT-}\lambda$. Assume $t, u \in \texttt{CT-}\lambda$. We define $t{\mapsto}u$ as: for some $t' \in \texttt{lab-CT-}\lambda$ such that $\overline{t'} = t$, some $0 \notin X \subseteq N$, if $u' = \phi_X(t')$ then $u = \overline{u'}$.

We prove that $\mapsto$ is confluent. Assume that $t, u, v \in \texttt{CT-}\lambda$, $t{\mapsto}u$ and $t{\mapsto}v$, in order to prove that $u, v{\mapsto}w$ for some $w \in \texttt{CT-}\lambda$. By definition we have:

1. for some $t'_1 \in \texttt{lab-CT-}\lambda$ such that $\overline{t'_1} = t$, some $0 \notin X \subseteq N$, if $u' = \phi_X(t'_1)$ then $u = \overline{u'}$.
2. for some $t'_2 \in \texttt{lab-CT-}\lambda$ such that $\overline{t'_2} = t$, some $0 \notin Y \subseteq N$, if $u' = \phi_Y(t'_2)$ then $u = \overline{u'}$.

We relabel t'_1 and t'_2 by replacing all $i \notin\in X \cup Y$ with 0, all $i \in X \setminus Y$ with 1, all $i \in X \cap Y$ with 2, all $i \in Y \setminus X$ with 3. The result is some term $t'_3 \in \texttt{lab-CT-}\lambda$, such that $t = \overline{t'_3}$, and if $u'' = \phi_{\{1,2\}^*}(t'_3)$ then $\overline{u''} = \overline{\phi_X(t'_1)} = \overline{u'} = u$ and and if $v'' = \phi_{\{2,3\}^*}(t'_3)$ then $\overline{v''}) = \overline{\phi_Y(t'_2)} = \overline{v'} = v$. We have $0 \notin \{1,2\} \cup \{2,3\}$.

By Lemma 9.13 the term $w' = \phi_{\{1,2,3\}^*}(t'_3) \in \texttt{lab-CT-}\lambda$ is the common value of $\phi_{\{2,3\}^*}(\phi_{\{1,2\}^*}(t'_3))$ and of $\phi_{\{1,2\}^*}(\phi_{\{2,3\}^*}(t'_3))$. Let $w = \overline{w'}$. Then $u, v{\mapsto}w$ by definition, as we wished to show.

10 An older proof of Church-Rosser in the limit

In order to recover Church-Rosser we have to consider a more general notion of reduction $\mapsto_X$, which allow to reduce *infinitely many redexes in one step*: all those in a *decidable* set X of redexes of t, and possibly all current redexes This reduction can generate new redexes of course.

We will prove that $\mapsto_X$ is confluent: namely, we will prove that if $t \mapsto_X u$ and $t \mapsto_Y v$, then for some w, Z, T we will have $u \mapsto_Z w$ and $v \mapsto_T w$. We call this property Infinite Church-Rosser. Infinite Church-Rosser will imply the unicity of the safe part of the safe normal form.

Our first problem is that infinite reductions can easily loop, therefore Infinite Church-Rosser is stated for the set $\mathbb{\Lambda}_\perp$ of terms with possibly *undefinite* subterms. This is not an obstacle for the goals of this paper, as we will see.

We formally state Infinite Church-Rosser as follows. "*For all $t, u, v \in \mathbb{\Lambda}_\perp$, all decidable sets X, Y of redexes of t, if $t \mapsto_X u$ and $t \mapsto_Y v$, then for some $w \in \mathbb{\Lambda}_\perp$, some decidable set Z, T of redexes of u, v we have $u \mapsto_Z w$ and $v \mapsto_T w$ and $t \mapsto_{X \cup Y} w$*".

We represent a decidable set X of positions of redexes, and in fact any decidable set of sub-terms by a map $\phi_X : |t| \to \{\text{true}, \text{false}\}$ such that $l \in X$ if and only if $\phi_X(l) = \text{true}$. Our first step is to precise how ϕ_X changes when redexes in X are moved or duplicated.

Definition 10.1 (Substitution, subterms and labels) *Suppose $t, u \in \mathbb{\Lambda}_\perp$ and X is a decidable set of redexes of t and Y a decidable set of redexes of u.*

1. *$Z = [Y/x]$ is a a decidable set of redexes of $t[u/x]$ defined as: for all $l \in |t|$, $m|u|$: if l is a free occurrence of x we set $\phi_Z(l \star m) = \phi_Y(m)$, otherwise $\phi_Z(l \star m) = \text{false}$.*
2. *$X[Y/x] = X \cup [Y/x]$.*
3. *If $n = 0, 1, 2$ and $t = c(t_1) \dots (t_n)$ and X a decidable set of redexes of t, then for all $1 \leq i \leq n$ we define a set X_i of redexes in t_i by $\phi_{X_i}(l) = \phi_X((i) \star l)$.*
4. *If $n = 0, 1, 2$ and $t = c(t_1) \dots (t_n)$ and for all $1 \leq i \leq n$ X_i is a decidable set of redexes of t_i, then we define a set X of redexes in t by $\phi_X(\mathtt{nil}) = \text{false}$ and $\phi_X((i) \star l) = \phi_{X_i}(l)$.*

We define the *unique* $u \in \mathbb{\Lambda}_\perp$ such that $t \mapsto_X u$ as the limit of a map $\rho(t, X, n)$ for $n \to \infty$. $\rho(t, X, n)$ starts with the undefined value $\perp$, then either holds the value $\perp$ forever, or at some step the root of the term $\rho(t, X, n)$ becomes some constructor of $\mathbb{\Lambda}$ and never changes again. At each step, if t itself is a redex in the set X then we reduce it, if t is not a redex in X then we move to the subterms of t. In both case we update X accordingly to some set of labels X'. We update any other set Z of labels in t in the same way to some Z'. We introduce a map $\sigma(t, X, n, Z)$ computing the value for a set Z of redexes after n steps. In particular we have $X' = \sigma(t, X, 1, X)$.

Definition 10.2 (Infinite reductions) *Assume $t \in \mathbb{\Lambda}_\perp$ and X, Z are decidable sets of redexes of t. Let $X_1, Y_1, \dots$ as in Def. 10.1.*

We set $\rho(t, X, 0) = \perp$. If $\phi_X(t) = \text{true}$, then we set:

1. *If $t = (\lambda x^T.b)(a)$, then*
   (1) $\rho(t, X, n+1) = \rho(b[a/x], X', n)$
   (2) $\sigma(t, X, n+1, Z) = \sigma(b[a/x], X, n, Z')$
   where $Z' = (Z_1)_1[Z_2/x]$.
2. *If $t = \mathtt{cond}(f, g)(0)$, then*
   (1) $\rho(t, X, n+1) = \rho(f, X', n)$
   (2) $\sigma(t, X, n+1, Z) = \sigma(f, X, n, Z')$
   where $Z' = (Z_1)_1$.
3. *If $t = \mathtt{cond}(f, g)(\mathtt{S}(u))$, then*
   (1) $\rho(t, X, n+1) = \rho(g(u), X', n)$
   (2) $\sigma(t, X, n+1, Z) = \sigma(g(u), X, n, Z')$
   where $Z' = \mathtt{ap}((Z_1)_2, (Z_2)_1)$.

Assume $\phi_X(t) = \text{false}$ *and* $t = c(t_1)\dots(t_h)$ *for some* $h = 0, 1, 2$ *some* $t_1, \dots, t_h \in \mathbb{A}_\perp$. *Then we set*

1. $\rho(t, X, n+1) \quad = c(\rho(t_1, X_1, n)\dots(\rho(t_h, X_h, n))$
2. $\sigma(t, X, n+1, Z) = c(\sigma(t_1, X_1, n, Z_1)\dots\sigma(t_h, X_h, n, Z_h))$ *and* $Z' = c(Z_1)\dots(Z_h) = Z$.

We define $\rho(t, X) = \lim_{n\to\infty} \rho(t, X, n)$ *and* $\sigma(t, X, Z) = \lim_{n\to\infty} \sigma(t, X, n, Z)$ *and* $t \mapsto_X \rho(t, X)$.

Proposition 10.3 (Reduction and union) *Suppose* $t, u \in \mathbb{A}_\perp$ *and* X *are decidable sest of redexes of* t *and* Y *are decidable sets of redexes of* u *and* $l \in |t|$. *Let* $X', Y', \dots$ *as in Def. 10.1.*

1. $[Z \cup T/x] = [Z/x] \cup [T/x]$
2. $(X \cup Y)[Z \cup T/x] = X[Z/x] \cup Y[T/x]$
3. $(X \cup Y)_l = X_l \cup Y_l$
4. $c(X_1 \cup Y_1)\dots(X_n \cup Y_n) = c(X_1)\dots(X_n) \cup c(Y_1)\dots(Y_n)$
5. $(X \cup Y)' = X' \cup Y'$

We will prove that for all $t, u, v \in \mathbb{A}_\perp$, all decidable sets X, Y of redexes of t we have $\rho(\rho(t, X), Z) = \rho(t, X \cup Y)$ for $Z = \sigma(t, X, Y)$.

We order $\mathbb{A}_\perp$ with $t \le u$ if and only if $|t| \subseteq |u|$ and for all $l \in |t|$ either l has label $\perp$ in $|t|$ or l has the same label in $|t|$ and $|u|$.

We prove $\rho(\rho(t, X), Z) \le \rho(t, X \cup Y)$ (left-to-right implication) and $\rho(t, X \cup Y) \le \rho(\rho(t, X), Z)$ (right-to-left implication).

Lemma 10.4 (Infinite Church-Rosser (left-to-right)) *For all* $t, u, v \in \mathbb{A}_\perp$, *all decidable sets* X, Y *of redexes of* t: $\rho(\rho(t, X), Z) \le \rho(t, X \cup Y)$ *for* $Z = \sigma(t, X, Y)$.

Proof We have to prove that $\rho(\rho(t, X), Z) = \rho(t, X \cup Y)$ for $Z = \sigma(t, X, Y)$.

Let us abbreviate $Y_{(n)} = \sigma(t, X, n, Y)$ and $Y_{(\omega)} = \lim_{n\to\infty} Y_{(n)} = Z$. then $\rho(\rho(t, X), Z) = \rho(\rho(t, X), Y_{(\omega)}))$ is the limit of $\rho(\rho(t, X, n), Y_{(n)}, m)$ for $n, m \to \infty$.

We prove that for all $n, m \in N$ there is some $p \in N$ such that $\rho(\rho(t, X, n), Y_{(n)}, m) \le \rho(t, X \cup Y, p)$, and conversely that for all $p \in N$ there are $n, m \in N$ such that $\rho(t, X \cup Y, p) \le \rho(\rho(t, X, n), Y_{(n)}, m)$. It will follow that if a node is defined in $\rho(\rho(t, X), Z)$ then it is defined in $\rho(t, X \cup Y)$ and with the same constructor, and conversely.

$\rho(\rho(t, X, n), Y_{(n)}, m) = \perp$ we are done, suppose $\rho(\rho(t, X, n), Y_{(n)}, m) > \perp$. If $\phi_X(t) = \text{true}$, then we have:

1. If $t = (\lambda x^T.b)(a)$, then $\rho(t, X, n) = \rho(b[a/x], X', n-1)$, and by induction hypothesis $\rho(\rho(t, X, n), Y_{(n)}, m) = \rho(\rho(b[a/x], X', n-1), Y'_{(n-1)}, m) \le \rho(b[a/x], X' \cup Y', p) = \rho(b[a/x], (X \cup Y)', p) = \rho((\lambda x^T.b)(a), X \cup Y, p+1)$.
2. If $t = \mathtt{cond}(f, g)(0)$, then $\rho(t, X, n+1) = \rho(f, X', n)$, and by induction hypothesis $\rho(\rho(t, X, n), Y_{(n)}, m) = \rho(\rho(f, X', n-1), Y'_{(n-1)}, m) \le \rho(f, X' \cup Y, p) = \rho(f, X' \cup Y', p) = \rho(f, (X \cup Y)', p) = \rho(t, X \cup Y, p+1)$.
3. If $t = \mathtt{cond}(f, g)(\mathtt{S}(u))$, then $\rho(t, X, n+1) = \rho(g(u), X', n)$ and by induction hypothesis $\rho(\rho(t, X, n), Y_{(n)}, m) = \rho(\rho(f, X', n-1), Y'_{(n-1)}, m) \le \rho(f, X' \cup Y', p) = \rho(f, (X \cup Y)', p) = \rho(t, X \cup Y, p+1)$.

Assume $\phi_X(t) = \text{false}$. Then we have $\rho(t, X, n) = c(\rho(t_1, X_1, n-1)\dots(\rho(t_h, X_h, n-1))$ and $X = c(X_1)\dots c(X_h)$.

Suppose $\phi_Y(t) = \text{true}$.

1. If $t = (\lambda x^T.b)(a)$, $\rho(t, X, n) = (\lambda x^T.b')(a')$ then $\rho((\lambda x^T.b')(a'), Y_{(n)}, m) = \rho(b'[a'/x], Y'_{(n-1)}, n-1)$, and by induction hypothesis $\rho(b'[a'/x], Y'_{(n-1)}, n-1) \le \rho(b'[a'/x], X \cup Y', p) = \rho(b[a/x], X' \cup Y', p) = \rho(b[a/x], (X \cup Y)', p) = \rho((\lambda x^T.b)(a), X \cup Y, p+1)$.
2. If $t = \mathtt{cond}(f, g)(0)$, $\rho(t, X, n) = \mathtt{cond}(f', g')(0)$ then $\rho(t, X, n) = \rho(f, X', n-1)$, and by induction hypothesis $\rho(\rho(t, X, n), Y_{(n)}, m) = \rho(\rho(f, X', n-1), Y'_{(n-1)}, m) \le \rho(f, X' \cup Y, p) = \rho(f, X' \cup Y', p) = \rho(f, (X \cup Y)', p) = \rho(t, X \cup Y, p+1)$.

3. If $t = \mathtt{cond}(f,g)(\mathtt{S}(u))$, $\rho(t,X,n) = \mathtt{cond}(f',g')(\mathtt{S}(u'))$ then $\rho(t,X,n) = \rho(g(u),X',n-1)$ and by induction hypothesis $\rho(\rho(t,X,n),Y_{(n)},m) = \rho(\rho(g(u),X',n-1),Y'_{(n-1)},m) \leq \rho(g(u),X' \cup Y',p) = \rho(g(u),(X \cup Y)',p) = \rho(g(u),X \cup Y,p+1)$.

Assume $\phi_X(t) = \text{false}$ and $\phi_Y(t) = \text{false}$ and $t = c(t_1)\ldots(t_h)$ for some $h = 0,1,2$. Then $\rho(\rho(t,X,n),Y_{(n)},m) = c(\rho(\rho(t_1,X_1,n-1),(Y_{(n)})_1,m-1),\ldots,\rho(\rho(t_h,X_h,n-1),(Y_{(n)})_h,m-1) = c(\rho(\rho(t_1,X_1,n),(Y_1)_{(n-1)},m),\ldots,\rho(\rho(t_h,X_h,n),(Y_h)_{(n-1)},m) \leq c(\rho(t_1,X_1 \cup Y_1,p_1),\ldots,\rho(t_h,X_h \cup Y_h,p_h) = c(\rho(t_1,(X \cup Y)_1,p_1),\ldots,\rho(t_h,(X \cup Y)_h,p_h) \leq c(\rho(t_1,(X \cup Y)_1,p),\ldots,\rho(t_h,(X \cup Y)_h,p) = \rho(t,X \cup Y,p)$ for $p = \max(p_1,\ldots,p_h)$.

The proof of the opposite implication is similar.

Lemma 10.5 (Infinite Church-Rosser (right-to-left)) *For all $t,u,v \in \mathbb{A}_\perp$, all decidable sets X, Y of redexes of t: $\rho(t,X \cup Y) \leq \rho(\rho(t,X),Z)$ for $Z = \sigma(t,X,Y)$.*

Proof .....

Theorem 10.6 (Infinite Church-Rosser) *For all $t,u,v \in \mathbb{A}_\perp$, all decidable sets X, Y of redexes of t:*

1. *if $t \mapsto_X u$ and $t \mapsto_{X \cup Y} w$ and $Z = \sigma(t,X,Y)$ then $u \mapsto_Z w$.*
2. *if $t \mapsto_X u$ and $u \mapsto_Y v$, then for some $w \in \mathbb{A}_\perp$, for some decidable sets Z, T of redexes of u,v we have $u \mapsto_Z w$ and $v \mapsto_T w$.*

Proof 1. Assume if $t \mapsto_X u$ and $t \mapsto_{X \cup Y} w$ and $Z = \sigma(t,X,Y)$, in order to prove $u \mapsto_Z w$. Then $u = \rho(t,X)$ and $w = \rho(t,X \cup Y)$ and we have to prove $\rho(u,Z) = w$. This follows from $\rho(\rho(t,X),Z) = \rho(t,X \cup Y)$ (lemmas and).

2. Assume $t \mapsto_X u$ and $u \mapsto_Y v$. There is some $w \in \mathbb{A}_\perp$ such that $t \mapsto_{X \cup Y} w$. From the previous point we have $u \mapsto_Z w$ for $Z = \sigma(t,X,Y)$ and $v \mapsto_T w$ for $T = \sigma(t,Y,X)$

We define the safe trunk of a term as the part of the term which we can normalize with safe reductions only. In the rest of this section we will prove that Infinite Church-Rosser implies that if the safe trunk exists then it is unique.

Infinite Church-Rosser and Safe strong Normalization together imply that after finitely many steps all safe reductions reach the same safe trunk.

Definition 10.7 (Safe Trunk of a term) *Assume $t \in \mathbb{A}$.*

1. *The safe trunk of t is any expression $u[\mathtt{cond}(f_1,\cdot),\ldots,\mathtt{cond}(f_n,\cdot)]$ such that for some $g_1,\ldots,g_n$ we have $v = u[\mathtt{cond}(f_1,g_1),\ldots,\mathtt{cond}(f_n,g_n)]$ safe normal and $t \mapsto v$.*
2. *t is finite for safe reduction if and only if all infinite reduction sequences from t include only finitely many "safe" reduction steps.*

Lemma 10.8 (Safe Trunk of a term) *Assume t is finite for safe reductions. If the safe-trunk of t exists then it is unique.*

Proof Assume t is finite for safe reductions in order to prove that the safe-trunk of t is unique.

Assume that $u[\mathtt{cond}(f_1,\cdot),\ldots,\mathtt{cond}(f_n,\cdot)]$ and $u'[\mathtt{cond}(f'_1,\cdot),\ldots,\mathtt{cond}(f'_{n'},\cdot)]$ are safe-trunks for t, in order to prove that $u = u'$ and $n = n'$.

Then for some $g_1,\ldots,g_n$ and some $g'_1,\ldots,g'_n$ we have that $v = u[\mathtt{cond}(f_1,g_1),\ldots,\mathtt{cond}(f_n,g_n)]$ and $v' = u'[\mathtt{cond}(f'_1,g'_1),\ldots,\mathtt{cond}(f'_n,g'_{n'})]$ are safe-normal forms of t and all cond-expressions shown are maximal. The decomposition of each safe-normal form v is therefore unique: if $v = u"[\mathtt{cond}(f"_1,g"_1),\ldots,\mathtt{cond}(f"_n,g"_{n"})]$ then $u = u"$ and $n = n"$ and $f_1 = f"_1$, ..., $f_n = f"_n$. Each reduction from v, v' takes place in some $g_1,\ldots,g'_{n'}$.

By Infinite Church-Rosser (theorem 10.6) we deduce that v and v' are confluent outside the right-hand-side of cond-expressions, therefore for some $v"$ we have $v \mapsto_Z v"$ and $v' \mapsto_T v'"$. Since the reductions on v,v' take place in some $g_1,\ldots,g'_{n'}$, we deduce that $v" = u[\mathtt{cond}(f_1,g"_1),\ldots,\mathtt{cond}(f_n,g"_n)]$ and $v'" = u[\mathtt{cond}(f'_1,g'"_1),\ldots,\mathtt{cond}(f'_n,g'"_{n'})]$ for some $g"_1,\ldots,g'"_{n'}$. From the unicity of the decomposition of $v"$ with maximal cond-subterms we conclude that $u = u'$ and $n = n'$ and and $f_1 = f'_1$, ..., $f_n = f'_n$, as wished.

References

[1] A Circular Version of Gödel's T and its abstraction complexity. Anupam Das. arXiv:2012.14421v2 [cs.LO] 16 Jan 2021.

11 Appendix: Equivalence Between Cyclic and non-Cyclic System T

In this section we prove that the two systems $\mathbb{T}$ and $\mathtt{CT}$-λ are each interpretable into the other. Both interpretations preserve the reduction relation, applications, 0 and $\mathtt{S}$ and contexts.

The formal definition of the system $\mathbb{T}$ is as follows:

Types (denoted by $A, B, \ldots$) of $\mathbb{T}$ are inductively defined by N and $A \to B$.

Terms (denoted by $t, u, \ldots$) of $\mathbb{T}$ are inductively defined by x^A, $\lambda x^A.t$, tu, 0, $\mathtt{S}(t)$, $\mathtt{Rec}(t,u)$, and $\mathtt{ifz}(t,u)$.

Sequents is $\Gamma \vdash t : A$, where Γ is a context $x_1 : A_1, \ldots, x_n : A_n$.

Typing rules are defined as follows:

1. The $\mathtt{var}$ rule: $\Gamma \vdash x : A$, where $x : A \in \Gamma$.
2. The $\mathtt{weak}$ rule: $\Gamma \vdash t : A$ and $\Gamma \subsetneq \Gamma'$, where $\Gamma' \vdash t : A$.
3. The λ rule: $\Gamma, x : A \vdash t : B$ implies $\Gamma \vdash \lambda x.t : A \to B$.
4. The $\mathtt{ap}$ rule: $\Gamma \vdash t : A \to B$ and $\Gamma \vdash u : A$ implies $\Gamma \vdash tu : B$.
5. The 0 rule: $\Gamma \vdash 0 : N$.
6. The $\mathtt{S}$ rule: $\Gamma \vdash t : N$ implies $\Gamma \vdash \mathtt{S}(t) : N$.
7. The $\mathtt{Rec}$ rule: $\Gamma \vdash a : A$ and $\Gamma \vdash h : A, N \to A$ implies $\Gamma \vdash \mathtt{Rec}(a,h) : N \to A$.
8. The $\mathtt{ifz}$ rule: $\Gamma \vdash u : A$ and $\Gamma \vdash v : A$ implies $\Gamma \vdash \mathtt{ifz}(u,v) : N \to A$.

The basic reduction relation $\mapsto_0$ is defined by:

1. $(\lambda x.t)u \mapsto_0 t[u/x]$,
2. $\mathtt{Rec}(a,h)0 \mapsto_0 a$,
3. $\mathtt{Rec}(a,h)\mathtt{S}(u) \mapsto_0 h(\mathtt{Rec}(a,h)u))u$,
4. $\mathtt{ifz}(u,v)0 \mapsto_0 u$,
5. $\mathtt{ifz}(u,v)\mathtt{S}(u) \mapsto_0 v$,

Then the reduction relation $\mapsto_{\mathbb{T}}$ is the coontext and transitive closure of $\mapsto_0$.

We first provide the interpretation $(-)^\circ$ from $\mathbb{T}$ to $\mathtt{CT}$-λ that is inductively defined as follows.

$$(x^A)^\circ = x^A, \qquad (\lambda x^A.t)^\circ = \lambda x^A.(t)^\circ, \qquad (tu)^\circ = (t)^\circ(u)^\circ, \qquad (0)^\circ = 0, \qquad (\mathtt{S}(t))^\circ = \mathtt{S}((t)^\circ),$$
$$(\mathtt{ifz}(u,v))^\circ = \mathtt{cond}((u)^\circ, \lambda z.(v)^\circ), \text{ where } z \notin \mathrm{FV}(v), \text{ and}$$
$$(\mathtt{Rec}(a,h))^\circ = f, \text{ where } f = \mathtt{cond}((a)^\circ, \lambda n.(h)^\circ(fn)n).$$

Note that the translated terms are regular. We can easiliy check that the translation preserves substitution, namely $(t[u/x])^\circ = (t)^\circ[(u)^\circ/x]$.

The interpretation preserves typablity.

Proposition 11.1 *$\Gamma \vdash t : A$ holds in $\mathbb{T}$ implies $\Gamma \vdash (t)^\circ : A$ holds in $\mathtt{CT}$-λ.*

Proof It is shown by induction on the derivation of $\Gamma \vdash t : A$. We check only the two cases: the last rule is $\mathtt{ifz}$ or $\mathtt{Rec}$.

The case of $\mathtt{ifz}$: the sequent $\Gamma \vdash \mathtt{ifz}(u,v) : N \to A$ is obtained from subproofs of $\Gamma \vdash u : A$ and $\Gamma \vdash v : A$. Then, by the induction hypothesis, we have proofs of $\Gamma \vdash (u)^\circ : A$ and $\Gamma \vdash (v)^\circ : A$ in $\mathtt{CT}$-λ. Hence we have a proof of $\Gamma \vdash \mathtt{cond}((u)^\circ, \lambda z.(v)^\circ) : N \to A$ by $\mathtt{weak}$ and $\mathtt{cond}$, where z is a fresh variable. This is a proof of $\Gamma \vdash (\mathtt{ifz}(u,v))^\circ : N \to A$ in $\mathtt{CT}$-λ. Note that this is an expected proof, since it satisfies the global trace condition and regularity.

The case of $t = \mathtt{Rec}(a,h)$: the sequent $\Gamma \vdash \mathtt{Rec}(a,h) : N \to A$ is obtained from subproofs of $\Gamma \vdash a : A$ and $\Gamma \vdash h : A, N \to A$. By the definition, $(\mathtt{Rec}(a,h))^\circ = \mathtt{cond}((a)^\circ, \lambda n.(h)^\circ((\mathtt{Rec}(a,h))^\circ n)n)$ holds. Then we have the following proof in $\mathtt{CT}$-λ.

$$\dfrac{\dfrac{\dfrac{\vdots\ \Pi_h}{\Gamma \vdash (h)^\circ : A, N \to A} \quad \dfrac{\Gamma \vdash (\mathtt{Rec}(a,h))^\circ : \mathbf{N} \to A \quad (\dagger)}{\Gamma, n : N \vdash (\mathtt{Rec}(a,h))^\circ : \mathbf{N} \to A}}{\Gamma, n : N \vdash (h)^\circ : A, N \to A} \quad \dfrac{\Gamma, n : \mathbf{N} \vdash (\mathtt{Rec}(a,h))^\circ n : A}{\Gamma, n : \mathbf{N} \vdash (h)^\circ((\mathtt{Rec}(a,h))^\circ n) : N \to A}\ \mathtt{ap}_v}{\dfrac{\Gamma, n : \mathbf{N} \vdash (h)^\circ((\mathtt{Rec}(a,h))^\circ n)n : A}{\Gamma \vdash \lambda n.(h)^\circ((\mathtt{Rec}(a,h))^\circ n)n : \mathbf{N} \to A}\ \mathtt{ap}_v} \quad \dfrac{\vdots\ \Pi_a}{\Gamma \vdash (a)^\circ : A}}{\Gamma \vdash (\mathtt{Rec}(a,h))^\circ : \mathbf{N} \to A \quad (\dagger)}\ \mathtt{cond}$$

This proof satisfies the global trace condition: Its infinite path is either one that eventually an infinite path of Π_a, one that eventually an infinite path of Π_h, or one that loops between the upper (†) and the lower (†). Then paths of the former two cases contains a progressing trace by the induction hypothesis, and the last case contains a progressing trace (the trace consists of the red Ns). Hence we obtain a proof of $\Gamma \vdash (\mathtt{Rec}(a,h))^\circ : N \to A$ as we wished.

For considering the corresponding reduction relation of $\mathbb{T}$ and the safe reduction of CT-λ, we need to introduce a reduction strategy, namely the call-by-name strategy, that prevents reductions of terms inside `Rec` and `ifz` that require to reduce terms inside `cond` of translated CT-λ.

Definition 11.2 *The call-by-name reduction relation $\mapsto_n$ is inductively defined as follows:*

1. *If $t \mapsto_0 t'$, then $t \mapsto_n u$,*
2. *$t \mapsto_n t'$ implies $tu \mapsto_n t'u$,*
3. *$t \mapsto_n t'$ implies $\mathtt{Rec}(a,h)t \mapsto_n \mathtt{Rec}(a,h)t'$, and*
4. *$t \mapsto_n t'$ implies $\mathtt{ifz}(u,v)t \mapsto_n \mathtt{ifz}(u,v)t'$.*

The call-by-name reduction and the

Lemma 11.3 *Assume that t has a closed term of type N. Then the following claims hold.*

1. *If t is not a numeral, then $t \mapsto_n u$ for some u.*
2. *$t \mapsto n$ if and only-if $t \mapsto_n n$.*

Proof The claim 1. is shown by induction on t. By the assumption, t has a form of either $\mathtt{S}(t')$, $(\lambda x.t_1)t_2\vec{t}$, $\mathtt{Rec}(a,h)t'\vec{t}$, or $\mathtt{ifz}(v_1,v_2)t'\vec{t}$. For the first case, there is u' such that $t \mapsto_n u'$ by the induction hypothesis. Then, take $u = \mathtt{S}(u')$. For the second case, take $u = t_1[t_2/x]\vec{t}$. For the third case, take $u = a\vec{t}$ if $t' = 0$, take $u = h(\mathtt{Rec}(a,h)t'')t''\vec{t}$ if $t' = \mathtt{S}(t'')$, and take $\mathtt{Rec}(a,h)u'\vec{t}$ otherwise, where $t' \mapsto_n u'$ by the induction hypothesis. For the last case, take $u = v_1\vec{t}$ if $t' = 0$, take $u = v_2\vec{t}$ if $t' = \mathtt{S}(t'')$, and take $\mathtt{ifz}(v_1,v_2)u'\vec{t}$ otherwise, where $t' \mapsto_n u'$ by the induction hypothesis.

We show the claim 2.. The right-to-left direction is trivially shown. The left-to-right direction is shown by 1. and the confluency of $\mapsto$: Assume that $t \mapsto n$. By the first claim, $t \mapsto_n n'$ for some numeral n'. Then we have $n = n'$ by the confluency. Hence $t \mapsto_n n$ holds as we wished.

Proposition 11.4 *Let t and u be terms of $\mathbb{T}$.*

1. *$t \mapsto_n u$ implies $(t)^\circ \mapsto_{\mathtt{safe}} (u)^\circ$.*
2. *If t is a closed term of type N and $t \mapsto n$, then $(t)^\circ \mapsto_{\mathtt{safe}} n$.*

Proof The point 1. is shown by induction on $\mapsto_n$. It is enough to check the base cases.

The case of $(\lambda x.t)u \mapsto_n t[u/x]$ is shown by $((\lambda x.t)u)^\circ = (\lambda x.(t)^\circ)(u)^\circ \mapsto_{\mathtt{safe}} (t)^\circ[(u)^\circ/x] = (t[u/x])^\circ$.

The case of $\mathtt{Rec}(a,h)0 \mapsto_n a$ is shown by $(\mathtt{Rec}(a,h)0)^\circ = \mathtt{cond}((a)^\circ, \lambda n.(h)^\circ((\mathtt{Rec}(a,h))^\circ n)n)0 \mapsto_{\mathtt{safe}} (a)^\circ$.

The case of $\mathtt{Rec}(a,h)\mathtt{S}(u) \mapsto_n h(\mathtt{Rec}(a,h)u)u$ is shown by

$$\begin{aligned}(\mathtt{Rec}(a,h)\mathtt{S}(u))^\circ = \mathtt{cond}((a)^\circ, \lambda n.(h)^\circ((\mathtt{Rec}(a,h))^\circ n)n)\mathtt{S}((u)^\circ) &\mapsto_{\mathtt{safe}} (\lambda n.(h)^\circ((\mathtt{Rec}(a,h))^\circ n)n)(u)^\circ \\ &\mapsto_{\mathtt{safe}} (h)^\circ((\mathtt{Rec}(a,h))^\circ(u)^\circ)(u)^\circ) = (h(\mathtt{Rec}(a,h)u)u)^\circ\end{aligned}$$

The case of $\mathtt{ifz}(u,v)0 \mapsto_n u$ is shown by $(\mathtt{ifz}(u,v)0)^\circ = \mathtt{cond}((u)^\circ, \lambda z.(v)^\circ)0 \mapsto_{\mathtt{safe}} (u)^\circ$.

The case of $\mathtt{ifz}(u,v)\mathtt{S}(t) \mapsto_n u$ is shown by $(\mathtt{ifz}(u,v)\mathtt{S}(t))^\circ = \mathtt{cond}((u)^\circ, \lambda z.(v)^\circ)\mathtt{S}(t) \mapsto_{\mathtt{safe}} (\lambda z.(v)^\circ)t \mapsto_{\mathtt{safe}} (v)^\circ$.

Hence we have the point 1.. The point 2. is obtained by the point 1. and Lemma 11.3 2..

The rest of the section if but an early draft The opposite interpretation, from CT-λ to $\mathbb{T}$, it has been defined by proof-theory for the combinatorial version of circular $\mathbb{T}$. For CT-λ, instead we will define an algorithm taking an infinite term in CT-λ, described as a finite circular tree, and returning a term in $\mathbb{T}$ with the required properties.

First, we need a notion of confluence and of extensional equality for functionals of CT-λ and of $\mathbb{T}$.

We define $t \sim_{\beta,\mathtt{rec}} u$ (a syntactical confluence for $\mathbb{T}$) if and and only if $t, u \in \mathbb{T}$ and t, u have the same type and for some $v \in \mathbb{T}$: $t \mapsto_{\beta,\mathtt{rec}} v$ and $u \mapsto_{\beta,\mathtt{rec}} v$.

We define an equivalence relation (an extensional equality for $\mathbb{T}$) $\sim_{\mathbb{T}}$ on $\mathbb{T}$, by induction on the type.

Definition 11.5 (An extensional equality on $\mathbb{T}$) *Assume $t, u \in \mathbb{T}$.*

1. *If $t, u : N$ and t, u are closed then we set: $(t \sim_{\mathbb{T}} u) \Leftrightarrow (t \sim_{\beta,\mathtt{rec}} u)$.*
2. *If $t, u : A, \vec{A} \to N$ and t, u are closed then we set: $(t \sim_{\mathbb{T}} u) \Leftrightarrow \forall a \in \mathbb{T}.(a : A), (a \text{ closed}) \Rightarrow (t(a) \sim_{\mathbb{T}} u(a))$.*
3. *If $t, u : \vec{A} \to N$ and $\mathrm{FV}(t), \mathrm{FV}(u) \subseteq \vec{x}$ then we set:*

$$(t \sim_{\mathbb{T}} u) \Leftrightarrow \forall \vec{a} \in \mathbb{T}.(\vec{a} : \vec{A}), (\vec{a} \text{ closed}) \Rightarrow (t[\vec{a}/\vec{x}] \sim_{\mathbb{T}} u[\vec{a}/\vec{x}])$$

Then we can consider $\mathbb{T}$ as a structure $(\mathbb{T}/\sim_{\mathbb{T}}, 0, \mathtt{S}, \mathtt{ap})$ with natural numbers, functionals and extensional equality. In the same way we define an equivalence relation on terms of CT-λ which denote functionals. We only consider terms whose type and context only include the atomic type N, and no type variables.

Definition 11.6 (An extensional equality on CT-λ) *Assume $t, u \in$ CT-λ.*

1. *If $t, u : N$ and t, u are closed then we set: $(t \sim_{\mathtt{CT}\text{-}\lambda} u) \Leftrightarrow (t \sim_{\mathtt{CT}\text{-}\lambda} u)$.*
2. *If $t, u : A, \vec{A} \to N$ and t, u are closed then we set: $(t \sim_{\mathtt{CT}\text{-}\lambda} u) \Leftrightarrow \forall a \in \mathtt{CT}\text{-}\lambda.(a : A), (a \text{ closed}) \Rightarrow (t(a) \sim_{\mathtt{CT}\text{-}\lambda} u(a))$.*
3. *If $t, u : \vec{A} \to N$ and $\mathrm{FV}(t), \mathrm{FV}(u) \subseteq \vec{x}$ then we set: $(t \sim_{\mathtt{CT}\text{-}\lambda} u) \Leftrightarrow \forall \vec{a} \in \mathtt{CT}\text{-}\lambda.(\vec{a} : \vec{A}), (\vec{a} \text{ closed}) \Rightarrow (t[\vec{a}/\vec{x}] \sim_{\mathtt{CT}\text{-}\lambda} u[\vec{a}/\vec{x}])$.*

Our goal is now to prove that there is an embedding from the types of N-functionals in $(\mathtt{CT}\text{-}\lambda/\sim_{\mathtt{CT}\text{-}\lambda}, 0, \mathtt{S}, \mathtt{ap})$ to the entire $(\mathbb{T}/\sim_{\mathbb{T}}, 0, \mathtt{S}, \mathtt{ap})$. We ignore types of CT-$\lambda$ including type variables, they have no corresponding in $\mathbb{T}$.

For each N-functional type T we have to define a map ϕ_T from terms of type T of CT-λ to terms of type T of $\mathbb{T}$. We abbreviate ϕ_T with ϕ and we require:

1. If $t \sim_{\mathtt{CT}\text{-}\lambda} u$ then $\phi(t) \sim_{\mathbb{T}} \phi(u)$
2. If $t : A \to B$, $u : A$ then $\phi(t(u)) \sim_{\mathbb{T}} \phi(t)(\phi(u))$
3. $\phi(0) \sim_{\mathbb{T}} 0$ and $\phi(\mathtt{S}(t)) \sim_{\mathbb{T}} \mathtt{S}(\phi(t))$

We will define an embedding from CT-λ to $\mathbb{T}$ extended with $+, \times$-types. There is an embedding from $\mathbb{T}$ extended with $+, \times$-types to $\mathbb{T}$ with only $N, \to$, therefore it is enough to embed CT-λ to $\mathbb{T}$ extended with $+, \times$-types.

We suppose be fixed a cyclic λ-term $t \in \mathtt{CT}\text{-}\lambda$, $t : T$, and we use several ingredients. First we consider all $\Gamma_i \vdash u_i : U_i$ for u_i sub-term of t, we turn it into a simple type $\Gamma_i \to U_i$, then we consider a single type $S = \Sigma_i \Gamma_i \to U_i$.

1. We suppose be given a map $\mathtt{trunk}_t : N \to S = \Sigma_i \Gamma_i \to U_i$, such that $\mathtt{trunk}_t(n)$ is the unfolding of t with all subterms $u : U$ in the level number n replaced by a dummy term 0_U. We use the type S in order to have a common type and context for all sub-terms u.
2. For each pair of subterms u, v of t and each path π from u to v. we suppose be given a one-to-many trace relation R_π between the indexes of N-arguments of u and of the N-arguments of v. We close the set of relations R_π by composition and we obtain a finite set (of exponential size in t).

By the global trace condition for each infinite composition of R_π there is some infinitely progressing trace. This means that there is some index i in the domain of R_π such that for some $n \in N$ we have $R_\pi^n(i,i)$ and the variable i progresses.

We consider any assignment $t' \equiv t[\vec{n},\vec{x}](\vec{m},\vec{y})$ of the arguments of t. The idea is to find map ϕ such that for all $p \geq \vec{n},\vec{n}$ we have $\texttt{trunk}_{t'}(m)$ stationary for all $m \geq \phi(p)$, therefore $t' = \texttt{trunk}_{t'}(\phi(p))$.

$\phi_c(p)$ is the map computing the maximum number of nodes for an Erdos tree in c colors with height $\leq p$ and branching $\leq c$ (there is at most one child per color). Whenever an Erdos tree has at least a branch of length $cp+1$ and c colors, then we have a monotonically colored sequence of length $cp+1$, therefore at least one homogeneous set of length $p+1$. If we take any composition of $\phi_c(p)+1$ times some relations R_π, there is some homogeneous set of $(p+1)$ elements decorated with a single R_π, and for some i some variable starting from some value $\leq p$ and decreasing p times.

This means that each branch terminates and the whole computation of $\texttt{trunk}_{t'}(\phi(p))$ terminates.

For a c-color tree, the value of $\phi_c(p)$ is $1+c+c^2+\ldots+c^{p-1} = (c^p-1)/(c-1)$. Suppose $f : N \to N$ is some weakly increasing map. We want to prove that there is some Erdos tree including some branch including some f-homogeneous set: some homogeneous set with first point h followed by $f(h)$ more points. We want to define a functional in $\mathbb{T}$ such that all Erdos trees with $\geq F(f)$ nodes include some f-homogeneous set.

We call a Ramsey functional any functional $F(\vec{x},c)$ associated to e any functional taking weakly increasing functionals $\vec{x}$, and returning an upper bound for the size of a c-color Erdos tree including some homogeneous set with first node l with value $\leq e(\vec{x},\vec{F},l)$, followed by $\leq e(\vec{x},\vec{F},l)$ more nodes, in which some variable i decreases by 1 each n steps (n number of sub-terms of the cyclic term).

We require that e is any primitive recursive functional, that is, e is defined by simply typed lambda calculus plus recursion on $\texttt{Seq}(N)$, and $\vec{F}$ are Ramsey functionals.

We claim that all sub-terms of t have a computation time bounded by some Ramsey functional. If this is not the case, we define some infinite path with a infinitely progressing trace associated to some infinitely decreasing numeral, contradiction.

.....