On Long Normal Inhabitants of a Type

SABINE BRODA and LUÍS DAMAS, *DCC & LIACC, Universidade do Porto, Portugal.*
Email: {sbb,luis}@ncc.up.pt

Abstract

In this paper we give a complete, formal definition of the formula-tree proof method, prove its correctness and illustrate its adequateness for research in the area of inhabitation of simple types.

Keywords: Lambda calculus, simple types, normal inhabitants, natural deduction, normal proofs.

1 Introduction

Inhabitation of types in the simply typed λ-calculus has been a major topic of study over the years. This problem is equivalent to the problem of provability of formulas in the implicational fragment of propositional intuitionistic logic. In fact, implicational formulas and simple types are syntactically identical and by the Curry–Howard isomorphism, cf. [18], every inhabitant of a type τ, i.e. every λ-term to which type τ can be assigned, may be regarded as a proof of the formula τ. In this paper we generally use λ-terms as a representation for proofs.

Research carried out in this area led to a large number of results on several topics, including proof-generating and counting algorithms [22, 11, 4, 5], [6] and complexity results [24, 17]. Also a number of sufficient and/or necessary conditions on the uniqueness of normal inhabitants/proofs were obtained [19, 15, 20, 23, 16, 1, 2]. But, in spite of the number of results that have been achieved, there still seems to be some lack of intuition on the subject. In fact, most of the results do not seem to be obvious at first sight and their proofs are frequently very complex and laborious, making them difficult to follow.

In [7] we introduced, in a rather informal way, a new method for investigating inhabitation. Here, types were given an alternative representation by splitting them into atomic parts and the process of constructing inhabitants consisted of combining these parts, following a set of rules determined by the structure of the formula. This simplification of the process of constructing inhabitants has provided us with a lot of insight into the relationship of the structure of types and their normal inhabitants. Since then, we have been using the method repeatedly for recognizing and proving new results and simplifying the proofs of others, cf. [8, 9, 10], and believe it to be a valuable tool for research in this area.

In this paper we will, in contrast with [7], finally provide a complete, formal description of the formula-tree proof method, prove its correctness and illustrate its adequateness for research in this area with several applications. In Section 2 we recall some results on simple types and their normal inhabitants, motivating the formalization of the formula-tree proof method given in Section 3. Several applications regarding inhabitation and provability in intuitionistic logic are respectively given in Section 4 and Section 5. We show the potential of the method in identifying and proving new results and simplifying the proofs of previous ones.

Vol. 15 No. 3, © The Author, 2005. Published by Oxford University Press. All rights reserved.
For Permissions, please email: journals.permissions@oupjournals.org
doi:10.1093/logcom/exi016

2 Preliminaries

We assume familiarity with the simply typed λ-calculus as described in [14] or [3]. Our notation differs from those two, since we denote type-variables (atoms) by 'A,B,C,...' and arbitrary types by lower-case Greek letters. For type-assignment we consider the system $\mathbf{TA}_\lambda$ of simply typed λ-calculus à la Curry (for an introduction see [14] or [3]). Occasionally we will also use typed terms, as described in Chapter 5 of [14], as a codification of $\mathbf{TA}_\lambda$-deductions. For typed terms note that given a β-normal inhabitant M of a type τ, there is exactly one $\mathbf{TA}_\lambda$-deduction that assigns the type τ to M. In this unique deduction of the formula $\vdash M : \tau$, every variable and subterm is given a type and the result of decorating all variables and subterms in M with the type given to them during this deduction is called a *typed term* and denoted by M^τ. The result of erasing the types in M^τ is denoted by $M^{\not\tau}$ and is clearly equal to M. For an introduction on typed terms see Chapter 5 of [14].

EXAMPLE 2.1
The typed inhabitant M^τ of $\tau = (A \to B) \to A \to B$, where $M = \lambda xy.xy$ is

$$M^\tau = (\lambda x^{A\to B} y^A.(x^{A\to B} y^A)^B)^{(A\to B)\to A\to B}.$$

However, when writing typed terms we will sometimes omit some of the types. As such, we might just write $M^\tau = \lambda x^{A\to B} y^A.xy$, $\lambda xy.x^{A\to B}y$ or even $\lambda xy.xy$ if convenient.

In the following we will recall some of the less standard results on simple types and their normal inhabitants. These are partly due to Ben-Yelles, cf. [4], and an exhaustive exposition can be found in [14]. Following the notation in [14], we will write $\underline{o}$ when referring to a particular occurrence of an object o.

2.1 Long normal inhabitants and their η-families

A β-normal inhabitant M of a type τ is called a *long* normal inhabitant of τ iff every variable-occurrence z in M is followed by the longest sequence of arguments allowed by its type, i.e. iff each component with form $(zP_1 \ldots P_n)$, $(n \geq 0)$ that is not in a function position has atomic type. The finite set of all terms obtained by η-reducing a λ-term M is called the *η-family* of M and denoted by $\{M\}_\eta$. It has been shown [4, 14] that the η-families of the long normal inhabitants of τ partition the set of normal inhabitants of τ, denoted by $\mathtt{Nhabs}(\tau)$, into non-overlapping finite subsets, each η-family containing just one long member. Furthermore, Ben-Yelles, cf. [4, 14], showed that every normal inhabitant of a type τ can be η-expanded to one unique (up to α-conversion) long normal inhabitant of τ. A simple expansion-algorithm can be found in [14]. Thus, when studying normal inhabitants of a type one might focus on the set of its long normal inhabitants from which all normal inhabitants can be obtained by η-reduction.

2.2 Subpremisses

Every type τ can be uniquely written as

$$\tau = \tau_1 \to \ldots \tau_n \to A$$

with $n \geq 0$. The type-variable A is called the *tail* of τ and denoted by $\mathtt{tail}(\tau)$. Furthermore, if $n \geq 1$, then $\tau_1, \ldots, \tau_n$ are called its *arguments*. In the following we recall the notion of subpremiss of a type [14]. Note that each subpremiss is a particular occurrence of a subtype and that not all occurrences of subtypes are subpremisses. As in [14] we will write $\underline{o}$ when referring to a specific occurrence of an object o.

DEFINITION 2.2
Occurrences of subtypes of a type τ satisfying the following are called negative (resp. positive) *subpremisses* of τ as follows.

- τ is a positive subpremiss of τ;
- if $\tau = \tau_1 \to \ldots \to \tau_n \to A$, then every positive (resp. negative) subpremiss of any of τ_j, $1 \leq j \leq n$, is a negative (resp. positive) subpremiss of τ.

EXAMPLE 2.3
The subpremisses of the type

$$\tau = ((A \to B) \to A \to B) \to (A \to B) \to A \to B$$

are all underlined occurrences of subtypes in

$$\underline{(\underline{(\underline{\underline{A} \to B}) \to \underline{A} \to B}) \to (\underline{\underline{A} \to B}) \to \underline{A} \to B}.$$

Note, that the number of lines under the (complete) occurrence of a subtype is odd if it is a positive subpremiss and even if it is a negative subpremiss.

A term M has a *bound-variable clash* iff M contains an abstractor $\underline{\lambda x}$ and a (free, bound or binding) occurrence of x that is not in its scope. On the other hand, every term can be α-converted to a term without bound-variable clashes.

If M is a long normal inhabitant of τ without bound-variable clashes and x a variable in M, then all occurrences of x in M^τ are decorated with the same type. Furthermore, there is exactly one negative subpremiss $\underline{\tau_x}$ of τ such that τ_x decorates all occurrences of x in M^τ. This particular occurrence of τ_x can be computed by the following and will be denoted by $\mathtt{nsp}(x, M^\tau)$.

DEFINITION 2.4
Consider a typed long normal inhabitant without bound-variable clashes
$M^\tau = \lambda x_1^{\tau_1} \ldots x_n^{\tau_1}.(y N_1^{\sigma_1} \ldots N_m^{\sigma_m})^A$ of type $\tau = \tau_1 \to \ldots \to \tau_n \to A$. We define $\mathtt{nsp}(x, M^\tau)$ by the following.

- If $x = x_i$ for some $1 \leq i \leq n$, then $\mathtt{nsp}(x, M^\tau)$ is the occurrence $\underline{\tau_i}$ in $\tau = \tau_1 \to \ldots \to \tau_{i-1} \to \underline{\tau_i} \to \tau_{i+1} \to \ldots \to \tau_n \to A$;
- otherwise, x occurs in exactly one N_i, $1 \leq i \leq m$. In this case, let $\mathtt{nsp}(x, M^\tau) = \mathtt{nsp}(x, N_i^{\sigma_i})$.

This procedure terminates clearly. On the other hand, it is easy to show by induction on the number of iterations that for every variable x in M, $\underline{\tau_x} = \mathtt{nsp}(x, M^\tau)$ is a negative subpremiss of τ.

2.3 Tree-domains and labelled trees

We will use ι to range over finite, possibly empty, sequences of positive natural numbers $\langle i_1, ..., i_n \rangle$, $\iota \cdot \iota'$ to denote the concatenation of two sequences and $|\iota|$ to denote the length of a sequence. If i is a natural number we will write $\iota \cdot i$ instead of $\iota \cdot \langle i \rangle$ and $i \cdot \iota$ instead of $\langle i \rangle \cdot \iota$. We will say that ι' is an *initial segment* of ι and write $\iota' \preceq \iota$ iff there exists ι'' such that $\iota = \iota' \cdot \iota''$.

We recall the following formal definitions about labelled trees, cf. [13].

DEFINITION 2.5
A *tree-domain* is a nonempty set $D_{\mathbf{t}}$ of sequences of natural numbers such that

- if $\iota \in D_{\mathbf{t}}$ and $\iota' \preceq \iota$ then $\iota' \in D_{\mathbf{t}}$;
- if $\iota \cdot i \in D_{\mathbf{t}}$ then $\iota \cdot j \in D_{\mathbf{t}}$ for every j such that $1 \leq j < i$.

DEFINITION 2.6
An $\mathcal{L}$*-labelled tree* $\mathbf{t}$ is a map from some tree domain $D_{\mathbf{t}}$ into a set $\mathcal{L}$. A tree $\mathbf{t}$ is *finite* iff $D_{\mathbf{t}}$ is finite.

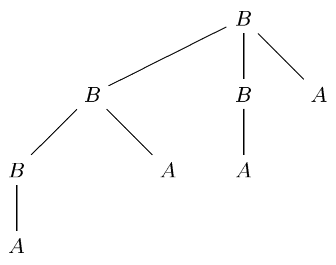
EXAMPLE 2.7
For $D_{\mathbf{t}} = \{\langle\rangle, \langle 1\rangle, \langle 2\rangle, \langle 3\rangle, \langle 1,1\rangle, \langle 1,2\rangle, \langle 1,1,1\rangle, \langle 2,1\rangle\}$ and $\mathcal{L} = \{A, B\}$ the tree $\mathbf{t}$ defined by

$$\mathbf{t}(\langle\rangle) = \mathbf{t}(\langle 1\rangle) = \mathbf{t}(\langle 1,1\rangle) = \mathbf{t}(\langle 2\rangle) = B$$

and

$$\mathbf{t}(\langle 1,1,1\rangle) = \mathbf{t}(\langle 1,2\rangle) = \mathbf{t}(\langle 2,1\rangle) = \mathbf{t}(\langle 3\rangle) = A$$

can also be represented as follows.



We will call elements of $\mathcal{L}$ *labels* of the tree $\mathbf{t}$, and elements of $D_{\mathbf{t}}$ *nodes* of the tree. A node ι is a *leaf* node iff $\iota \cdot 1 \notin D_{\mathbf{t}}$. An *internal* node is a node which is not a leaf node. A node ι is a *descendant* of a node ι' iff $\iota' \prec \iota$. A node ι is a *direct descendant* of a node ι' iff $\iota' = \iota \cdot i$ for some i. A node ι' is a (direct) *ancestor* of a node ι iff ι is a (direct) descendant of ι'. The *height* of a finite tree $\mathbf{t}$ is $max_{\iota \in D_{\mathbf{t}}} |\iota|$.

DEFINITION 2.8
If $\mathbf{t}$ is a tree and ι is a node of $\mathbf{t}$, then the *subtree* $\mathbf{t}[\iota]$ of $\mathbf{t}$ at ι is the tree with domain $D_{\mathbf{t}[\iota]} = \{\iota' \mid \iota \cdot \iota' \in D_{\mathbf{t}}\}$ defined by $\mathbf{t}[\iota](\iota') = \mathbf{t}(\iota \cdot \iota')$.

DEFINITION 2.9
If $\mathbf{t_1}, ..., \mathbf{t_n}$ are $\mathcal{L}-$labelled trees and $l \in \mathcal{L}$ then let $l[\mathbf{t_1}, ..., \mathbf{t_n}]$ denote the tree with domain $\{\langle\rangle\} \cup \bigcup_{1 \leq i \leq n} \{i \cdot \iota \mid \iota \in D_{\mathbf{t_i}}\}$ defined by

$$\begin{array}{rcl} l[\mathbf{t_1}, ..., \mathbf{t_n}](\langle\rangle) & = & l \\ l[\mathbf{t_1}, ..., \mathbf{t_n}](i \cdot \iota) & = & \mathbf{t_i}(\iota) \qquad (\text{for } 1 \leq i \leq n). \end{array}$$

Note that any labelled tree can be written in a unique way as $l[\mathbf{t_1}, ..., \mathbf{t_n}]$ where $n \geq 0$.

EXAMPLE 2.10
The tree from example 2.7 can be written as

$$B[B[B[A[]], A[]], B[A[]], A[]].$$

Every β-normal term M is of the form $\lambda x_1 \dots x_n.yN_1 \dots N_m$, where $n, m \geq 0$ and such that $N_1, \dots, N_m$ are also terms in β-normal form. The variable y is called the *head* of M and is denoted by $\mathtt{head}(M)$. The Böhm tree[1] of M, denoted by $BT(M)$, is

$$\lambda x_1 \dots x_n.y$$

if $m = 0$ and otherwise

$$\begin{tikzcd}[arrows=-] & \lambda x_1 \dots x_n.y \arrow[dl] \arrow[dr] & \\ BT(N_1) & \cdots & BT(N_m) \end{tikzcd}.$$

EXAMPLE 2.11
For $D_{\mathbf{t}} = \{\langle\rangle, \langle 1\rangle, \langle 2\rangle, \langle 1,1\rangle\}$ and $\mathcal{L} = \{\lambda xyz.x, \lambda w.y, w, z\}$ the tree $\mathbf{t}$ defined by

$$\begin{aligned} \mathbf{t}(\langle\rangle) &= \lambda xyz.x \\ \mathbf{t}(\langle 1\rangle) &= \lambda w.y \\ \mathbf{t}(\langle 2\rangle) &= z \\ \mathbf{t}(\langle 1,1\rangle) &= w \end{aligned}$$

is the Böhm tree of the term $\lambda xyz.x(\lambda w.yw)z$ and can also be represented as follows.

$$\begin{tikzcd}[arrows=-] & \lambda xyz.x \arrow[dl] \arrow[dr] & \\ \lambda w.y \arrow[d] & & z \\ w & & \end{tikzcd}$$

2.4 *Total discharge convention*

In the simple type system with the total discharge convention, or Prawitz' natural deduction system [21], all term-variables have the same name and are indexed by their type. Clearly every inhabitant of a type τ in the total discharge convention is an inhabitant in $\mathbf{TA}_\lambda$ (with a great probability of bound-variable clashes). Furthermore, it has been shown [22] that for every inhabitant M of τ in $\mathbf{TA}_\lambda$ there exists an inhabitant $f(M)$ of τ in the total discharge convention system. This term $f(M)$ is obtained from M by renaming each variable v in M by x_{τ_v}, where $\underline{\tau_v} = \mathtt{nsp}(v, M^\tau)$. This renaming is not injective as can be seen in the following example.

EXAMPLE 2.12
Consider the type $\tau = A \to A \to A$ and its two normal inhabitants $\lambda xy.x$ and $\lambda xy.y$. The corresponding term in the system with the total discharge convention is in both cases $\lambda x_A x_A.x_A$.

On the other hand, given an inhabitant N of τ in the system with the total discharge convention one can compute the set $\mathcal{C}(N)$ of all inhabitants M of τ in $\mathbf{TA}_\lambda$ such that $f(M) = N$, with the following algorithm.

ALGORITHM $\mathcal{C}(\_)$ 2.13
Given a λ-term N with $n \geq 1$ differently indexed variables $x_{\alpha_1}, \dots, x_{\alpha_n}$, we define its expansion-set $\mathcal{C}(N) = N\,\underbrace{(0, \dots, 0)}_{n}$ as follows.[2]

[1]Note, that we restrict the definition of Böhm trees to the case of β-normal forms.
[2]The first case, i.e. $k_i = 0$, corresponds to free occurrences of variables in N.

$$\begin{array}{lll} x_{\alpha_i}(k_1,\ldots,k_n) & = \{x_{\alpha_i}\} & \text{if } k_i = 0 \\ x_{\alpha_i}(k_1,\ldots,k_n) & = \{x^0_{\alpha_i}, x^1_{\alpha_i}, x^2_{\alpha_i},\ldots,x^{k_i-1}_{\alpha_i}\}, & \text{if } k_i \geq 1 \\ (MN)(k_1,\ldots,k_n) & = \{M_iN_j \mid M_i \in M(k_1,\ldots,k_n), N_j \in N(k_1,\ldots,k_n)\} & \\ (\lambda x_{\alpha_i}.N)(k_1,\ldots,k_n) & = \{\lambda x^{k_i}_{\alpha_i}.N_i \mid N_i \in N(k_1,\ldots,k_i+1,\ldots,k_n)\} & \end{array}$$

EXAMPLE 2.14
For $N = \lambda x_A x_A.x_A$ from example 2.12 we have

$$\begin{array}{ll} \mathcal{C}(N) & = (\lambda x_A x_A.x_A)(0) \\ & = \{\lambda x^0_A.N_1 \mid N_1 \in (\lambda x_A.x_A)(1)\} \\ & = \{\lambda x^0_A x^1_A.N_1 \mid N_1 \in x_A(2)\} \\ & = \{\lambda x^0_A x^1_A.N_1 \mid N_1 \in \{x^0_A, x^1_A\}\} \\ & = \{\lambda x^0_A x^1_A.x^0_A, \lambda x^0_A x^1_A.x^1_A\} \\ & =_\alpha \{\lambda xy.x, \lambda xy.y\} \end{array}$$

The following proposition shows the correctness of the algorithm $\mathcal{C}(\_)$.

PROPOSITION 2.15
Let N be a normal λ-inhabitant of τ in the system with the total discharge convention having $n \geq 1$ differently indexed variables $x_{\alpha_1},\ldots,x_{\alpha_n}$. Then, the following hold.

1. $M \in \mathcal{C}(N)$ if and only if $f(M) = N$.
2. Every element of $\mathcal{C}(N)$ is an inhabitant of τ.
3. If N is long and $M \in \mathcal{C}(N)$, then M is also long.

PROOF. These properties result directly from the following facts. First, if $M \in \mathcal{C}(N)$ then M and N have exactly the same structure. Second, it is easy to show by induction on M and using Definition 2.4 that for every occurrence of a variable with type α in M the corresponding occurrence in N has name x_α. Finally, note that algorithm $\mathcal{C}(N)$ collects all terms whose structure is identical to the structure of N and in which all occurrences of variables have the same types as the corresponding occurrences in N. ■

COROLLARY 2.16
Let N be a normal λ-inhabitant of τ in the system with the total discharge convention and M a long normal inhabitant of τ. Then,

$$M \in \mathcal{C}(f(M)) \qquad \text{and} \qquad f(\mathcal{C}(N)) = N.$$

It follows from this last result that the expansion-sets of the long normal inhabitants of τ in the system with the total discharge convention partition the set of long normal inhabitants into disjoint, finite subsets. This is illustrated in Figure 1 where the small circles in the set of normal inhabitants represent the always finite η-families of $\texttt{Nhabs}(\tau)$, each of them containing exactly one long term. The big circles around the η-families represent the partition induced by the total discharge convention. Then, it follows from Proposition 2.15 that f is a bijective function from the set of these classes into the set of long normal terms in the total discharge convention system and that the inverse of f is $\mathcal{C}$.

So one may look at long normal inhabitants of τ in the system with the total discharge convention as a compact representation for finite sets of long normal inhabitants and their η-families. On the other hand, this 'compaction' may be too strong and unify too many long inhabitants with distinct

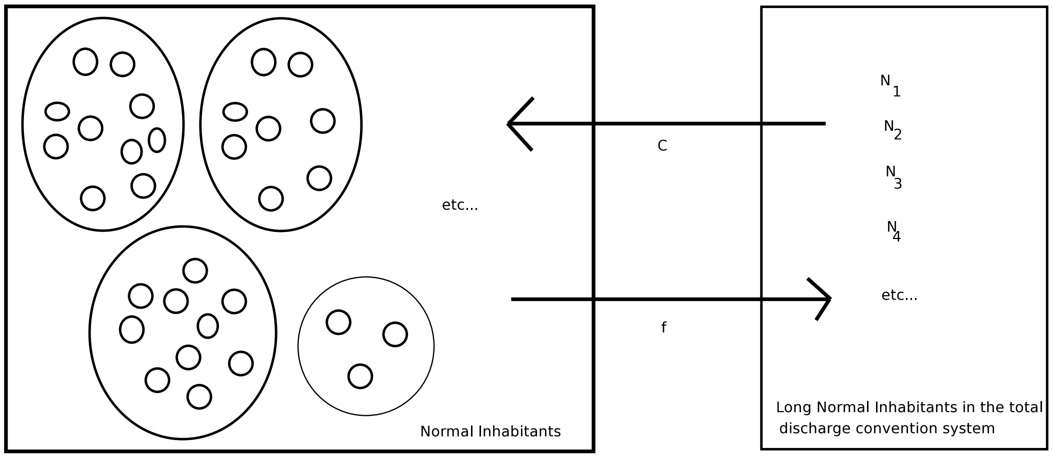

etc...
Normal Inhabitants
C
f
N1
N2
N3
N4
etc...
Long Normal Inhabitants in the total discharge convention system

FIGURE 1. Partition of $\mathtt{Nhabs}(\tau)$ by the total discharge convention

properties. For instance, note that the two long terms represented by $\lambda x_A x_A.x_A$ in the previous example do not have the same principal type and so we would like them to be represented by different terms. This will be achieved later in this paper, where we present another class of λ-terms which will be called terms in standard form. Similar to terms in the system with the total discharge convention, also every long standard inhabitant will stand for a finite set of long normal inhabitants, but now all with the same principal type.

3 The formula-tree proof method

3.1 The formula-tree of a type

We associate with any type τ a tree labelled by its type-variables, i.e. atomic types, as follows.

DEFINITION 3.1
Let τ be a type and $\mathcal{L}$ the set of atomic variables of τ, then the *formula-tree* FT_τ is an $\mathcal{L}$-labelled tree defined by the following.

- If τ is an atom A, then FT_A is $A[]$;
- if τ is of the form $\tau_1 \to \ldots \to \tau_n \to A$, then its formula-tree is $A[FT_{\tau_1}, ..., FT_{\tau_n}]$.

EXAMPLE 3.2
The formula-tree of the type

$$\tau = ((A \to B) \to A \to B) \to (A \to B) \to A \to B$$

is the tree $\mathbf{t}$ from example 2.7.

The converse notion of formula-tree is defined below.

DEFINITION 3.3
Given a finite tree $\mathbf{t}$ labelled by atomic types we associate a type $\widehat{\mathbf{t}}$ with $\mathbf{t}$ as follows:

1. $\widehat{A[]} = A$
2. $\widehat{A[\mathbf{t_1},...,\mathbf{t_n}]} = \widehat{\mathbf{t_1}} \to ... \to \widehat{\mathbf{t_n}} \to A$.

In the remainder of this paper we will use $\widetilde{\iota}_\tau$ as a shorthand for $\widehat{FT_\tau[\iota]}$. As such, $\widetilde{\iota}_\tau$ is the subtype of τ that corresponds to the subtree of FT_τ at node ι. The following properties of of $\widehat{FT_\tau[\iota]}$ follow easily from the definitions.

LEMMA 3.4
Consider a type τ and a node ι in FT_τ. Then,

1. $\widehat{FT_\tau} = \tau$;
2. $FT_\tau(\iota) = \mathtt{tail}(\widehat{FT_\tau[\iota]}) = \mathtt{tail}(\widetilde{\iota}_\tau)$;
3. if $\widehat{FT_\tau[\iota]} = \sigma_1 \to \ldots \to \sigma_m \to A$, then $\widehat{FT_\tau[\iota \cdot i]} = \sigma_i$;
4. if ι is even, then $\widehat{FT_\tau[\iota]}$ is a positive subpremiss of τ;
5. if ι is odd, then $\widehat{FT_\tau[\iota]}$ is a negative subpremiss of τ.

EXAMPLE 3.5
For τ from example 3.2 we have

$$\begin{array}{ll} \widetilde{\langle 1\rangle}_\tau & = (A \to B) \to A \to B \\ \widetilde{\langle 1,1\rangle}_\tau & = A \to B \\ \widetilde{\langle 1,2\rangle}_\tau & = A. \end{array}$$

Also,

$$\begin{array}{lll} FT_\tau(\langle 1\rangle) & = B & = \mathtt{tail}((A \to B) \to A \to B) \\ FT_\tau(\langle 1,1\rangle) & = B & = \mathtt{tail}(A \to B) \\ FT_\tau(\langle 1,2\rangle) & = A & = \mathtt{tail}(A). \end{array}$$

We will now associate with every variable x occurring in a term M^τ a unique node ι_x in the formula-tree FT_τ. For this purpose, we start by defining a function $\mathtt{Node}(\_, \_)$ that given type τ and a subpremiss $\underline{\sigma}$ of τ returns the position of $\underline{\sigma}$ in FT_τ. Then, ι_x will be defined as $\mathtt{Node}(\mathtt{nsp}(x, M^\tau), \tau)$.

DEFINITION 3.6
Given a type τ and a subpremiss $\underline{\sigma}$ of τ, we define $\mathtt{Node}(\underline{\sigma}, \tau)$ by the following.

- If $\underline{\sigma}$ is τ, then $\mathtt{Node}(\underline{\sigma}, \tau) = \langle\rangle$;
- if $\tau = \tau_1 \to \ldots \to \tau_n \to A$, the occurrence $\underline{\sigma}$ is in τ_i for some $1 \le i \le n$, and $\mathtt{Node}(\underline{\sigma}, \tau_i) = \iota$, then $\mathtt{Node}(\underline{\sigma}, \tau) = i \cdot \iota$.

Furthermore, if M is an inhabitant of τ and x a variable in M, then we will denote the position in τ of the negative subpremiss corresponding to x by ι_x. More formally, $\iota_x = \mathtt{Node}(\mathtt{nsp}(x, M^\tau), \tau)$.

EXAMPLE 3.7
All four occurrences of A in $\tau = ((A \to B) \to A \to B) \to (A \to B) \to A \to B$ are subpremisses of τ. The first and the last occurrences are negative subpremisses and the second and third occurrences are positive subpremisses.

For the first occurrence we have $\mathtt{Node}(\underline{A}, \tau) = \langle 1,1,1\rangle$, for the second $\mathtt{Node}(\underline{A}, \tau) = \langle 1,2\rangle$, for the third $\mathtt{Node}(\underline{A}, \tau) = \langle 2,1\rangle$ and finally for the fourth $\mathtt{Node}(\underline{A}, \tau) = \langle 3\rangle$.

The following result states the correctness of the function $\texttt{Node}$ and relates the sign (positive/negative) of a subpremiss with the parity of the length of its position.

LEMMA 3.8
Given a type τ and a subpremiss $\underline{\sigma}$ of τ, one has $\widetilde{\texttt{Node}(\underline{\sigma}, \tau)}_{\tau} = \sigma$. Furthermore, $|\texttt{Node}(\underline{\sigma}, \tau)|$ is even (resp. odd) if $\underline{\sigma}$ is a positive (resp. negative) subpremiss of τ.

PROOF. By induction on the length of $\texttt{Node}(\underline{\sigma}, \tau)$. If $\underline{\sigma}$ is τ, then $\underline{\sigma}$ is a positive subpremiss of τ and $\texttt{Node}(\underline{\sigma}, \tau) = \langle\rangle$ has even length 0. Furthermore, $FT_\tau[\langle\rangle] = FT_\tau$. Thus, $\widetilde{\texttt{Node}(\underline{\sigma}, \tau)}_{\tau} = \widehat{FT_\tau} = \tau = \sigma$.

Otherwise $\tau = \tau_1 \to \ldots \to \tau_n \to A$ and $\underline{\sigma}$ is a positive (resp. negative) subpremiss of τ_i, for some $1 \leq i \leq n$ and consequently a negative (resp. positive) subpremiss of τ. Furthermore, let $\texttt{Node}(\underline{\sigma}, \tau_i) = \iota$ and so $\texttt{Node}(\underline{\sigma}, \tau) = i \cdot \iota$. By the induction hypothesis, we conclude that $|\iota|$ is even (resp. odd), hence $|\texttt{Node}(\underline{\sigma}, \tau)| = |i \cdot \iota|$ is odd (resp. even). Moreover, $\widetilde{\iota}_{\tau_i} = \widehat{FT_{\tau_i}[\iota]} = \sigma$ and consequently $\widehat{FT_\tau[i \cdot \iota]} = \widehat{FT_{\tau_i}[\iota]} = \sigma$. ∎

LEMMA 3.9
If M is an inhabitant of τ and x a variable in M, then $FT_\tau(\iota_x) = \texttt{tail}(\sigma_x)$, where $\underline{\sigma_x} = \texttt{nsp}(x, M^\tau)$.

PROOF.

$$\begin{array}{lll} FT_\tau(\iota_x) & = \texttt{tail}(\widetilde{(\iota_x)}_\tau) & \text{Lemma 3.4} \\ & = \texttt{tail}(\widetilde{\texttt{Node}(\underline{\sigma_x}, \tau)}_\tau) & \text{Definition 3.6} \\ & = \texttt{tail}(\sigma_x) & \text{Lemma 3.8.} \end{array}$$

LEMMA 3.10
Let $(yN_1^{\sigma_1} \ldots N_m^{\sigma_m})^A$ be a subterm of M^τ such that $N_i = \lambda z_{i_1} \ldots z_{i_{k_i}}.P^{A_i}$ for $1 \leq i \leq m$ and $k_i \geq 0$. Then, $\iota_{z_{i_j}} = \iota_y \cdot \langle i, j\rangle$ for $1 \leq i \leq m$ and $1 \leq j \leq k_i$.

3.2 Long inhabitants in standard form

A long normal inhabitant M is said to be in *standard form* if all term-variables in M are of the form x_ι, such that ι is the position of the negative subpremiss in τ corresponding to x_ι. While standard long normal inhabitants are surely long normal inhabitants, the converse is not true. But again, there is a natural function f_s that transforms terms into standard form. In fact, if we define $f_s(M)$ as the term obtained from M by renaming each variable v in M by x_{ι_v}, then $f_s(M)$ is clearly in standard form.

EXAMPLE 3.11
Consider the again type $\tau = A \to A \to A$ and its two normal inhabitants $\lambda xy.x$ and $\lambda xy.y$ from Example 2.12. The formula tree of τ is

$$\begin{tikzcd} & A \arrow[dl, dash] \arrow[dr, dash] & \\ A & & A \end{tikzcd}$$

and $f_s(\lambda xy.x) = \lambda x_{\langle 1\rangle} x_{\langle 2\rangle}.x_{\langle 1\rangle}$, while $f_s(\lambda xy.y) = \lambda x_{\langle 1\rangle} x_{\langle 2\rangle}.x_{\langle 2\rangle}$.

Given a standard long normal inhabitant N of τ we can apply algorithm $\mathcal{C}(\_)$ from Definition 2.13 to N and it is easy to see that Proposition 2.15 and Corollary 2.16 still hold for this class of terms.

In fact, one could just repeat the proof substituting indexes α by indexes ι. So $\mathcal{C}(N)$ will again collect all long normal inhabitants M of τ such that $f_s(M) = N$ and consequently f_s induces a partition on the set of all long normal inhabitants of τ, each finite part being represented by exactly one standard long normal inhabitant. Note that $f_s(M_1) = f_s(M_2)$ clearly implies $f(M_1) = f(M_2)$, but the converse is not true as can be seen in the previous example. In Figure 2 this finer partition of $\mathtt{Nhabs}(\tau)$ is illustrated. Again, the small circles in $\mathtt{Nhabs}(\tau)$ represent the η-families and the outermost circles the partition induced by terms in the total discharge convention. Then, the circles in these classes and around the η-families are the expansion-sets of standard long normal inhabitants and represent thus the finer partition induced by these terms.

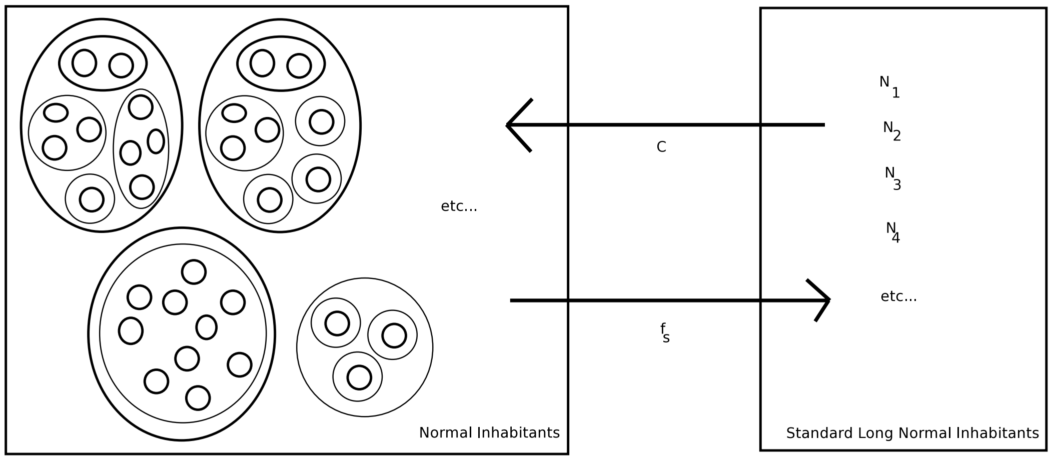


FIGURE 2. Partition of $\mathtt{Nhabs}(\tau)$ by standard forms

The next proposition, proved in appendix A, states an important property of the expansion-sets of standard long normal inhabitants.

PROPOSITION 3.12
Let $M_1, M_2 \in \mathcal{C}(N)$ for some standard long normal inhabitant N of τ. Then, M_1 and M_2 have the same principal type.

3.3 The proof-tree of an inhabitant

Let M be a closed long normal inhabitant of a type τ without bound-variable clashes. We want to associate with M a tree $\mathbf{t}_M$ labelled by nodes of FT_τ and want this tree to be a compact encoding for $f_s(M)$. We begin proving another property of standard long normal inhabitants, which essentially states that the abstraction sequences in these terms are redundant in the sense that they can be recovered from the type of the term and the head-variables in the term.

LEMMA 3.13
Let N_1 and N_2 be two standard long normal inhabitants of τ and let $bt(N_1)$ and $bt(N_2)$ be the result of erasing in the Böhm trees of N_1 and N_2 all abstraction sequences. If $bt(N_1) = bt(N_2)$, then $N_1 = N_2$.

PROOF. In order to prove that N_1 and N_2 have the same Böhm trees we show that for each node ι in $bt = bt(N_1) = bt(N_2)$ one has $BT(N_1)(\iota) = BT(N_2)(\iota)$. If $\tau = \tau_1 \to \ldots \to \tau_n \to A$ it follows from the definition of long standard terms that $BT(N_1)(\langle\rangle) = BT(N_2)(\langle\rangle) = \lambda x_{\langle 1\rangle} \ldots x_{\langle n\rangle}.bt(\langle\rangle)$.

Now consider a node $\iota \cdot i$ in bt and let $x_{\iota'} = bt(\iota)$. Then, $\widetilde{\iota'}_\tau$ is of the form $\sigma_1 \to \ldots \to \sigma_m \to A$ with $m \geq i$. Furthermore, if $\sigma_i = \sigma_{i_1} \to \ldots \to \sigma_{i_k} \to A_i$, then it follows again from N_1 and N_2 being long and standard that $BT(N_1)(\iota \cdot i) = BT(N_2)(\iota \cdot i) = \lambda x_{\iota' \cdot \langle i,1\rangle} \ldots x_{\iota' \cdot \langle i,k\rangle}.bt(\iota \cdot i)$. ∎

Considering this last result we define tree $\mathbf{t}_M$ as follows and call it the proof-tree of M.

DEFINITION 3.14
Consider a closed long normal inhabitant M of a type τ without bound-variable clashes. The *proof-tree* $\mathbf{t}_M$ is obtained from the Böhm tree $BT(M)$ by replacing every label of the form $\lambda x_1 \ldots x_n.y$ by the sequence/position ι_y.

Informally, the proof-tree of a long normal inhabitant M without bound-variable clashes is a tree which is isomorphic to the Böhm tree of M and such that every node labelled by some expression $\lambda x_1 \ldots x_n.y$ in the Böhm tree is labelled by $\iota_y = \texttt{Node}(\underline{\tau_y}, M^\tau)$ in $\mathbf{t}_M$, where $\underline{\tau_y} = \texttt{nsp}(y, M^\tau)$.

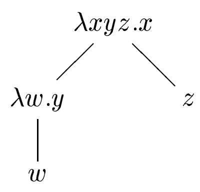
EXAMPLE 3.15
Consider the inhabitant

$$M = \lambda xyz.x(\lambda w.yw)z$$

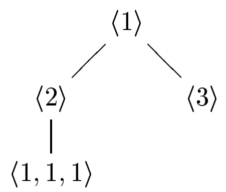
of

$$\tau = ((A \to B) \to A \to B) \to (A \to B) \to A \to B$$

from Example 2.7. In order to compute $\mathbf{t}_M$ note that $\iota_x = \langle 1\rangle$, $\iota_y = \langle 2\rangle$, $\iota_z = \langle 3\rangle$ and $\iota_w = \langle 1,1,1\rangle$, that the Böhm tree $BT(M)$ of M is



and that $f_s(M) = \lambda x_{\langle 1\rangle} x_{\langle 2\rangle} x_{\langle 3\rangle}.x_{\langle 1\rangle}(\lambda x_{\langle 1,1,1\rangle}.x_{\langle 2\rangle} x_{\langle 1,1,1\rangle}) x_{\langle 3\rangle}$. Thus, $\mathbf{t}_M =$



which can also be written as

$$\mathbf{t}_M = \langle 1\rangle[\langle 2\rangle[\langle 1,1,1\rangle[]], \langle 3\rangle[]].$$

We now prove some essential properties of proof-trees of long normal inhabitants. The first Lemma is straightforward and we need it for the proof of Theorem 3.17.

LEMMA 3.16
Let $M^\tau = \lambda x_1 \ldots x_n.P^A$ be a long normal inhabitant of $\tau = \tau_1 \to \ldots \to \tau_n \to A$ without bound-variable clashes and with proof-tree $\mathbf{t}_M$. Then, $\texttt{head}(M) = x_i$ and $\mathbf{t}_M(\langle\rangle) = \langle i\rangle$ for some $1 \leq i \leq n$. Furthermore, $|\iota_z| \geq 3$ for every variable z in M different from $x_1, \ldots, x_n$.

THEOREM 3.17
Let M be a closed long normal inhabitant of τ without bound-variable clashes and with proof-tree $\mathtt{t}_M$. Then,

1. $FT_\tau(\mathtt{t}_M(\langle\rangle)) = FT_\tau(\langle\rangle)$;
2. $\mathtt{t}_M(\iota)$ has odd length for any node ι of $\mathtt{t}_M$;
3. for any node ι of $\mathtt{t}_M$ the number of direct descendants of ι in $\mathtt{t}_M$ is exactly the same as the number of direct descendants of $FT_\tau(\mathtt{t}_M(\iota))$ and for any such descendant $\iota \cdot i$ we must have $FT_\tau(\mathtt{t}_M(\iota)\cdot i) = FT_\tau(\mathtt{t}_M(\iota \cdot i))$;
4. if $\mathtt{t}_M(\iota) = \iota' \cdot \langle i, j\rangle$ then there is some ancestor ι'' of ι in $\mathtt{t}_M$ such that $\mathtt{t}_M(\iota'') = \iota'$ and $\iota'' \cdot i$ is also an ancestor of ι or ι itself.

PROOF. 1. If $M^\tau = \lambda x_1 ... x_n . y^\sigma N_1 ... N_m$, then $\mathtt{tail}(\sigma) = \mathtt{tail}(\tau) = FT_\tau(\langle\rangle)$ and then the result follows from Lemma 3.9 and from $\mathtt{t}_M(\langle\rangle) = \iota_y$.

2. This follows from the fact that for every variable x in M the occurrence $\mathtt{nsp}(x, M^\tau)$ is a negative subpremiss of τ (cf. remark after Definition 2.4) and from Lemma 3.8.
3. If $\lambda x_1 \ldots x_n . y^{\sigma_y}$, with $\sigma_y = \sigma_1 \to \ldots \to \sigma_m \to A$ and $n \geq 0$, is the label of a node ι in $\mathtt{bt} = BT(M^\tau)$, then $\mathtt{bt}(\iota)$ has exactly m direct descendants $BT(N_1^{\sigma_1}), \ldots, BT(N_m^{\sigma_m})$ for some terms $N_1^{\sigma_1}, \ldots, N_m^{\sigma_m}$. Thus the same is true for $\mathtt{t}_M$. On the other hand, $\mathtt{t}_M(\iota) = \iota_y$, thus $\widehat{FT_\tau[\mathtt{t}_M(\iota)]} = \sigma_y$ and by Lemma 3.4, $\widehat{FT_\tau[\mathtt{t}_M(\iota)\cdot i]} = \sigma_i$ and consequently $FT_\tau(\mathtt{t}_M(\iota)\cdot i) = \mathtt{tail}(\sigma_i)$. Finally note that if z^{α_i} is the head of $N_i^{\sigma_i}$, then $\mathtt{tail}(\alpha_i) = \mathtt{tail}(\sigma_i)$. Thus, we conclude from Lemma 3.4 that $FT_\tau(\mathtt{t}_M(\iota \cdot i)) = FT_\tau(\iota_z) = \mathtt{tail}(\alpha_i) = \mathtt{tail}(\sigma_i)$.
4. If $\mathtt{t}_M(\iota) = \iota' \cdot \langle i, j\rangle$, then the node ι in $\mathtt{bt} = BT(M^\tau)$ is of the form $\lambda w_1 \ldots w_s . z$, such that $\iota_z = \iota' \cdot \langle i, j\rangle$, i.e. $\mathtt{bt}(\iota) = \lambda w_1 \ldots w_s . z$. We know from Lemma 3.16, that $\iota \neq \langle\rangle$ and also that z is not one of the variables in the initial abstraction sequence of M^τ. Thus, there is a subterm of M^τ of the form $\lambda u_1 \ldots u_t . (y N_1^{\sigma_1} \ldots N_{i'}^{\sigma_{i'}} \ldots N_m^{\sigma_m})^A$ such that $N_{i'} = \lambda z_1 \ldots z_k . v P_1 \ldots P_t$, where $k \geq 1$ and $z = z_{j'}$ for some $1 \leq j' \leq k$. Let ι'' be the node in $\mathtt{bt}$ such that $\mathtt{bt}(\iota'') = \lambda u_1 \ldots u_t . y$, thus $\mathtt{bt}(\iota'' \cdot i') = \lambda z_1 \ldots z_k . v$. It follows from Lemma 3.10 that $\iota_z = \iota_y \cdot \langle i, j\rangle$. Thus, $\iota' = \iota_y$, $i = i'$ and $j = j'$. On the other hand, the node ι labelled with $\lambda w_1 \ldots w_s . z$ occurs in the ith branch of subtree of $\mathtt{bt}$ rooted in node ι'' labelled with $\lambda u_1 \ldots u_t . y$. Thus, ι'' is clearly an ancestor of ι. Furthermore, if $\lambda z_1 \ldots z_k . v = \lambda w_1 \ldots w_s . z$, then $\iota = \iota'' \cdot i$, otherwise the node ι labelled with $\lambda w_1 \ldots w_s . z$ in $\mathtt{bt}$ is a descendant of the node $\iota'' \cdot i$ labelled with $\lambda z_1 \ldots z_k . v$ and consequently $\iota'' \cdot i$ is an ancestor of ι.

■

The converse of the above theorem is the following.

THEOREM 3.18
If $\mathtt{t}$ is a tree labelled by nodes of FT_τ satisfying the conditions of the previous theorem, then there exists a closed long standard inhabitant $N_{\mathtt{t}}$ of τ with proof-tree $\mathtt{t}_{N_{\mathtt{t}}} = \mathtt{t}$.

PROOF. Let $N_{\mathtt{t}}$ be the term whose Böhm tree BT, represented as a labelled tree, has domain $D_{\mathtt{t}}$ and is defined by

$\mathtt{BT}(\langle\rangle) = \lambda x_{\langle 1\rangle} \ldots x_{\langle k\rangle} . x_{\mathtt{t}(\langle\rangle)}$	where $\langle 1\rangle, \ldots, \langle k\rangle$ are all direct descendants of node $\langle\rangle$ in FT_τ
$\mathtt{BT}(\iota \cdot i) = \lambda x_{\mathtt{t}(\iota)\cdot\langle i,1\rangle} \ldots x_{\mathtt{t}(\iota)\cdot\langle i,k\rangle} . x_{\mathtt{t}(\iota\cdot i)}$	where $\mathtt{t}(\iota) \cdot \langle i, 1\rangle, \ldots, \mathtt{t}(\iota)\cdot\langle i, k\rangle$ are all direct descendants of node $\mathtt{t}(\iota)\cdot i$ in FT_τ.

We begin showing that $\texttt{BT}(\iota)$ is well defined. For this note that if ι is a node of $\mathbf{t}$, then there is a corresponding label $\mathbf{t}(\iota)$. So it remains to show that if $\iota \cdot i$ is a node in $\mathbf{t}$, then there is a node $\mathbf{t}(\iota) \cdot i$ in FT_τ. This follows from Condition 3 of Theorem 3.17. In fact, if $\iota \cdot i$ is a node in $\mathbf{t}$, then so is ι and $\mathbf{t}(\iota)$ is well defined and is by the hypothesis a node of FT_τ. It follows from Condition 3 that $FT_\tau(\mathbf{t}(\iota))$ has exactly as many direct descendants as $\mathbf{t}(\iota)$. Now, $\iota \cdot i$ is a descendant of node ι in $\mathbf{t}$ and since $D_\mathbf{t}$ is a tree-domain we conclude that ι has at least i direct descendants. So the same is true for node $\mathbf{t}(\iota)$ in FT_τ and consequently there is a node $\mathbf{t}(\iota) \cdot i$ in FT_τ. Finally, note that all indexes given to the variables in BT are nodes of FT_τ of odd length and correspond thus to negative subpremisses of τ, cf. Lemma 3.4.

$N_\mathbf{t}$ with Böhm tree BT is clearly a lambda-term. In order to show that $N_\mathbf{t}$ is closed, consider any node ι in BT. Then, $\texttt{BT}(\iota)$ is of the form $\lambda\vec{x}.x_{\mathbf{t}(\iota)}$ and we have to show that $x_{\mathbf{t}(\iota)}$ occurs in the abstraction sequence $\vec{x}$ or in an abstraction sequence of an ancestor of this node. Now, if $|\mathbf{t}(\iota)| = 1$, then $x_{\mathbf{t}(\iota)}$ occurs in the abstraction sequence of node $\langle\rangle$. Otherwise, $\mathbf{t}(\iota)$ is of the form $\iota' \cdot \langle i, j\rangle$ and by Condition 4 of Theorem 3.17 there is some ancestor ι'' of ι in $\mathbf{t}$ such that $\mathbf{t}(\iota'') = \iota'$ and $\iota'' \cdot i$ is also an ancestor of ι or ι itself. The abstraction sequence in this node $\iota'' \cdot i$ is by the definition of BT the sequence $\lambda x_{\mathbf{t}(\iota'')\cdot\langle i,1\rangle} \dots x_{\mathbf{t}(\iota'')\cdot\langle i,k\rangle} = \lambda x_{\iota'\cdot\langle i,1\rangle} \dots x_{\iota'\cdot\langle i,k\rangle}$, where $1, \dots, k$ are all existing nodes of the form $\iota' \cdot \langle i, m\rangle$ in FT_τ. So $\iota' \cdot \langle i, j\rangle$ has to be among them.

Finally, we have to show that $N_\mathbf{t}$ is a long inhabitant of τ, for then it is clearly standard. For this we begin noticing that for $\tau = \tau_1 \to \dots \to \tau_n \to A$ there is $\texttt{BT}(\langle\rangle) = \lambda x_{\langle 1\rangle} \dots \lambda x_{\langle n\rangle}.x_{\mathbf{t}(\langle\rangle)}$ and $\widetilde{\langle i\rangle}_\tau = \tau_i$ for $1 \leq i \leq n$. Furthermore, it follows from Condition 1. of Theorem 3.17 that $FT_\tau(\mathbf{t}(\langle\rangle)) = \texttt{tail}(\tau) = A$. Thus it suffices to show that every node ι in BT of the form $\lambda\vec{x}.x_{\mathbf{t}(\iota)}$, with $\widetilde{\mathbf{t}(\iota)}_\tau = \sigma_1 \to \dots \to \sigma_n \to A'$, has exactly n direct descendants in $\mathbf{t}$, and consequently in BT, that for any such descendant $\iota \cdot i$, if $\sigma_i = \sigma_{i_1} \to \dots \to \sigma_{i_k} \to A_i$, then $\texttt{BT}(\iota \cdot i) = \lambda x_{\mathbf{t}(\iota)\cdot\langle i,1\rangle} \dots x_{\mathbf{t}(\iota)\cdot\langle i,k\rangle}.x_{\mathbf{t}(\iota\cdot i)}$ and $FT_\tau(\mathbf{t}(\iota \cdot i)) = \texttt{tail}(\sigma_i)$. In fact, if $\widetilde{\mathbf{t}(\iota)}_\tau$ is of the form $\sigma_1 \to \dots \to \sigma_n \to A_i$, then $FT_\tau(\mathbf{t}(\iota)) = A_i$ and has exactly n direct descendants. So, the result follows from the definition of BT and from Condition 3. of Theorem 3.17. ∎

These last results show that in order to generate the normal inhabitants of a simple type τ one may compute the formula-tree of τ and then generate the trees labelled by its nodes that satisfy the conditions of Theorem 3.17. Then, the proof of Theorem 3.18 provides us with algorithm $\texttt{BT}(\_)$ to compute the corresponding standard long normal inhabitants of τ, from which the long normal inhabitants can be generated by algorithm $\mathcal{C}(\_)$.

This method is used by the *formula-tree proof method* which was first informally introduced in [7][3] and has already been successfully used in order to obtain results in the area of inhabitation and provability, [8, 9, 10]. In the formula-tree proof method the set of conditions in Theorem 3.17 have been reformulated into an equivalent, less formal set of conditions in order to obtain a neat and intuitive procedure of constructing proof-trees. In the following we will perform the reformulation of the conditions in Theorem 3.17, that have to be met by a tree $\mathbf{t}$ in order to correspond to a standard long normal inhabitant.

First note that the conditions on such a tree $\mathbf{t}$ can be resumed as follows.

(a) All nodes in $\mathbf{t}$ are labelled with nodes ι of odd length of FT_τ.

(b) If ι labels the root of $\mathbf{t}$, then node ι is labelled by the same type-variable $FT_\tau(\iota)$ in FT_τ as is the root $FT_\tau(\langle\rangle)$ of FT_τ.

[3]A short presentation of the method together with an implementation as a Java applet can be found at `http://www.ncc.up.pt/~sbb/FTLab/ftlab`

(c) If ι labels a node ι' in $\mathbf{t}$ and node ι in FT_τ has $n \geq 0$ direct descendants $\iota \cdot 1, \ldots, \iota \cdot n$, labelled respectively by type-variables $A_1, \ldots, A_n$, then also ι' has exactly n direct descendants $\iota' \cdot 1, \ldots, \iota' \cdot n$ in $\mathbf{t}$. Furthermore, the nodes labelling these n direct descendants in $\mathbf{t}$ are labelled in FT_τ by $A_1, \ldots, A_n$ respectively.

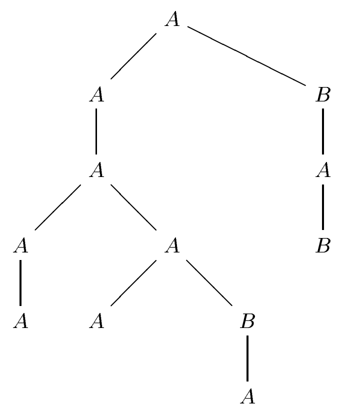
(d) Finally, if $\iota \cdot \langle i, j \rangle$ labels any node ι' in $\mathbf{t}$, then there is an ancestor ι'' of ι' in $\mathbf{t}$ labelled by ι and the node ι' is in the ith subtree rooted in ι''.

Since most of these conditions concern type-variables labelling the nodes of odd length in FT_τ and their direct descendants, it becomes natural to consider the parts of the formula-tree consisting of the odd nodes and their direct descendants and reformulate (a)–(d) as a set of conditions on these parts. As such, we will from now on consider a formula-tree FT_τ split into small parts p_ι, that we will call primitive parts, and which are formed by the nodes ι of odd length and their direct descendants. Additionally, we consider a root-part consisting of the root-node of FT_τ. In order to distinguish this part from the parts resulting from nodes with no direct descendants, we add an edge in the top of the type-variable in the former and an edge below the type-variables in the latter.

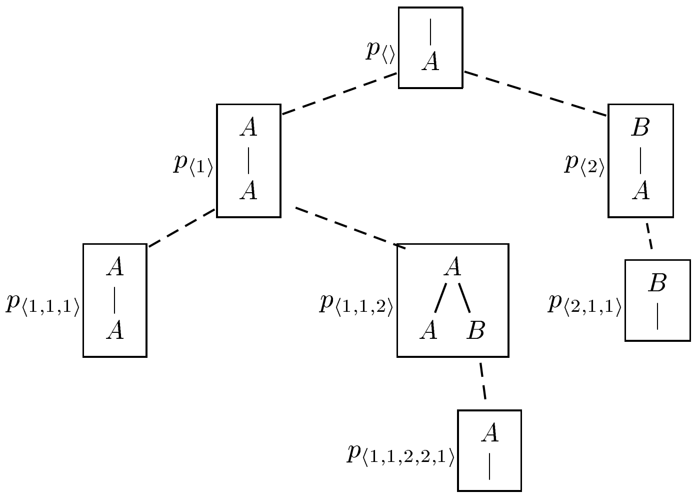
EXAMPLE 3.19
Consider the type

$$\tau = (((A \to A) \to (A \to (A \to B) \to A) \to A) \to A) \to ((B \to A) \to B) \to A$$

with formula-tree

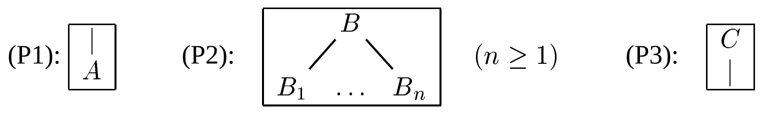


After splitting FT_τ into its primitive parts we obtain the following tree.



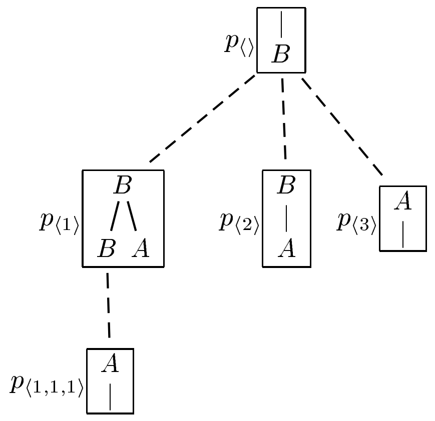
$p_{\langle\rangle}$
A
$p_{\langle 1\rangle}$
A
A
$p_{\langle 2\rangle}$
B
A
$p_{\langle 1,1,1\rangle}$
A
A
$p_{\langle 1,1,2\rangle}$
A
A B
$p_{\langle 2,1,1\rangle}$
B
$p_{\langle 1,1,2,2,1\rangle}$
A

We conclude that *primitive parts* are items of either one of the following forms (P1), (P2) or (P3), where $A, B, B_1, \ldots, B_n, C$ denote type-variables.



Here, $A, B_1, \ldots, B_n$ are called the *tail-variables* of the respective primitive part, while B and C are *head-variables*. The *arity* of a primitive part is the number of its tail-variables. Now, it is easy to see that after splitting a formula-tree FT_τ into primitive parts, it forms a tree-like structure consisting of labelled primitive parts. Here, the primitive part in the top is the only part of the form (P1) and has label $p_{\langle\rangle}$. Any other primitive part p_ι is of form (P2) or (P3), and ι is the position of its head-variable in FT_τ. Furthermore, p_ι descends directly from the ith tail-variable of another primitive part $p_{\iota'}$ of arity n, for some integers $0 < i \leq n$, and finally $\iota = \iota' \cdot \langle i, j\rangle$ for some $j \geq 1$.

EXAMPLE 3.20
Consider again type $\tau = ((A \to B) \to A \to B) \to (A \to B) \to A \to B$ from Example 3.15, whose formula-tree FT_τ, now split into primitive parts is the following.

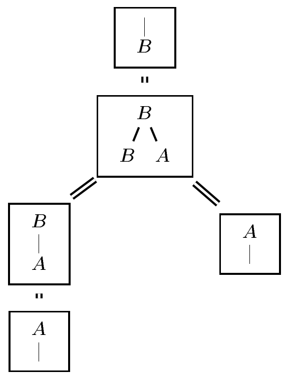


Consider again its long normal inhabitant $M = \lambda xyz.x(\lambda w.yw)z$ with proof-tree

$$\mathbf{t}_M = \begin{tikzcd} & \langle 1\rangle \arrow[dl, no head] \arrow[dr, no head] & \\ \langle 2\rangle \arrow[d, no head] & & \langle 3\rangle \\ \langle 1,1,1\rangle & & \end{tikzcd} .$$

We now replace in $\mathbf{t}_M$ every label ι by the primitive part p_ι and add a top-node $p_{\langle\rangle}$. Also, for every node labelled with a primitive part p_ι with $k \geq 1$ tail-variables and having $k \geq 1$ subtrees rooted in it, we consider for $1 \leq i \leq k$ the ith of these subtrees to be descending also from the ith tail-variable of p_ι. This is emphasized using double lines instead of simple ones. Then we obtain the following tree

$$\mathtt{PT}_{\mathbf{t}_M} = \begin{tikzcd} & p_{\langle 0\rangle} \arrow[d, equal] & \\ & p_{\langle 1\rangle} \arrow[dl, equal] \arrow[dr, equal] & \\ p_{\langle 2\rangle} \arrow[d, equal] & & p_{\langle 3\rangle} \\ p_{\langle 1,1,1\rangle} & & \end{tikzcd} =$$



.

One can recognize in this example that Conditions (a)–(d) above guarantee that a primitive part p_ι is linked to the tail-variable A of another primitive part only if p_ι has head-variable A. Also, if p_ι is a direct descendant of the kth tail-variable of another primitive part $p_{\iota'}$ in FT_τ, then every occurrence of p_ι in $\mathtt{PT}_{\mathbf{t}_M}$ must be in a subtree linked to the kth tail-variable of an occurrence of $p_{\iota'}$ (for an example where primitive parts occur more than once see Example 3.23).

DEFINITION 3.21
Given a type τ and a tree $\mathtt{PT}$ labelled by primitive parts of FT_τ and with top-node $p_{\langle\rangle}$, we denote by $\mathbf{t}_{\mathtt{PT}}$ the tree obtained from $\mathtt{PT}$ by deleting its top-node $p_{\langle\rangle}$ and substituting every node p_ι by ι.

Furthermore, if $\mathbf{t}$ is a tree labelled by nodes of odd length of FT_τ, let $\mathtt{PT}_{\mathbf{t}}$ be the tree obtained from $\mathbf{t}$ by substituting every node ι by p_ι and inserting a top-node $p_{\langle\rangle}$.

Now, consider a tree $\mathtt{PT}$ labelled with primitive parts of FT_τ and satisfying the following conditions.

i. $p_{\langle\rangle}$ labels the root of $\mathtt{PT}$.

ii. Every occurrence of a primitive part p_ι in $\mathtt{PT}$, with $n \geq 0$ tail-variables $A_1, \ldots, A_n$, has exactly n direct descendants $p_{\iota_1}, \ldots, p_{\iota_n}$ with head-variables $A_1, \ldots, A_n$ respectively.

iii. Finally, if p_ι labels a node ι' in $\mathtt{PT}$ and the primitive part p_ι descends directly from the ith tail-variable of another primitive part $p_{\iota''} \neq p_{\langle\rangle}$ in FT_τ, i.e. $\iota = \iota'' \cdot \langle i, j\rangle$, then there is an ancestor ι''' of ι' in $\mathtt{PT}$ labelled by p''_ι and the node ι' is in the ith subtree rooted in ι'''.

The following result states the equivalence of Conditions (a)–(d) and Conditions i–iii.

PROPOSITION 3.22
Consider a type τ and trees $\mathtt{PT}$ and $\mathbf{t}$, respectively satisfying Conditions i–iii and Conditions (a)–(d) above. Then, $\mathbf{t}_{\mathtt{PT}}$ and $\mathtt{PT}_{\mathbf{t}}$ satisfy respectively Conditions (a)–(d) and Conditions i–iii. Furthermore, $\mathtt{PT}_{\mathbf{t}_{\mathtt{PT}}} = \mathtt{PT}$ and $\mathbf{t}_{\mathtt{PT}_{\mathbf{t}}} = \mathbf{t}$.

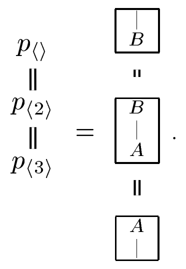
PROOF. We show that if $\mathtt{PT}$ satisfies Conditions i–iii, then $\mathbf{t}_{\mathtt{PT}}$ satisfies Conditions (a)–(d). The proof of the corresponding result for $\mathbf{t}$ and $\mathtt{PT}_{\mathbf{t}}$ is similar. First note that it follows from Definition 3.21 that ι labels a node ι' in $\mathbf{t}_{\mathtt{PT}}$ if and only if p_ι labels node $1 \cdot \iota'$ in $\mathtt{PT}$. Also, $\langle\rangle$ is the only node in $\mathtt{PT}$ labelled by $p_{\langle\rangle}$. Since all primitive parts in FT_τ, but $p_{\langle\rangle}$, are indexed by a sequence of odd length, we conclude that Condition (a) is met by $\mathbf{t}_{\mathtt{PT}}$. Now let ι be the label of the root of $\mathbf{t}_{\mathtt{PT}}$. Then, p_ι is the label of node $\langle 1\rangle$ in $\mathtt{PT}$, whose ancestor $\langle\rangle$ is labelled by $p_{\langle\rangle}$. Condition ii guarantees that the head-variable of p_ι equals the tail-variable of $p_{\langle\rangle}$. Thus, also Condition (b) is met by $\mathbf{t}_{\mathtt{PT}}$. The same is true for Conditions (c) and (d), since ii and iii are essentially transcriptions of them, in terms of primitive parts rather than sequences. Finally, $\mathtt{PT}_{\mathbf{t}_{\mathtt{PT}}} = \mathtt{PT}$ and $\mathbf{t}_{\mathtt{PT}_{\mathbf{t}}} = \mathbf{t}$ follow directly from the symmetry in Definition 3.21. ∎

From now on we will focus on trees labelled by primitive parts and satisfying Conditions i–iii above. In consequence of Proposition 3.22 and abusing on the notation, [4] we will also call them proof-trees.

EXAMPLE 3.23
Consider again the formula

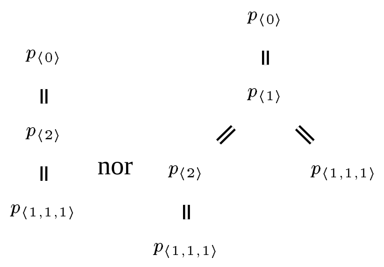
$$\tau = ((A \to B) \to A \to B) \to (A \to B) \to A \to B$$

with formula-tree FT_τ split into primitive parts as illustrated in Example 3.20. Another proof-tree for τ is

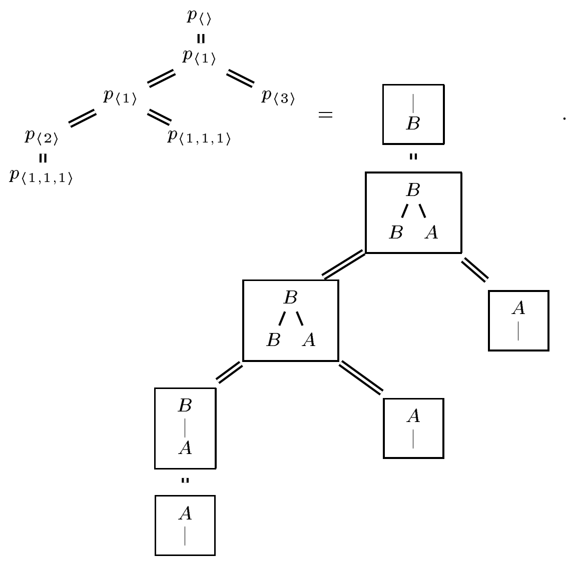


[4]Originally the term proof-tree referred to trees labelled by nodes of FT_τ and satisfying Conditions a–d.

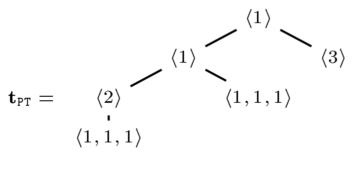
Note that neither



are proof-trees for τ, since they do not respect the hierarchy given by FT_{τ}, which requires that in a proof-tree all occurrences of $p_{\langle 1,1,1\rangle}$ occur in some subtree that is rooted in the first branch of an occurrence of $p_{\langle 1\rangle}$.
Another proof-tree for τ is



For this last proof-tree PT we have



$$N_{\mathbf{t}_{\text{PT}}} = \lambda x_{\langle 1\rangle} x_{\langle 2\rangle} x_{\langle 3\rangle} . x_{\langle 1\rangle} (\lambda x_{\langle 1,1,1\rangle} . x_{\langle 1\rangle} (\lambda x_{\langle 1,1,1\rangle} . x_{\langle 2\rangle} x_{\langle 1,1,1\rangle}) x_{\langle 1,1,1\rangle}) x_{\langle 3\rangle},$$

and

$$\mathcal{C}(N_{\mathbf{t}_{\text{PT}}}) =_{\alpha} \{\lambda xyz.x(\lambda u_1.x(\lambda u_2.yu_1)u_1)z,\ \lambda xyz.x(\lambda u_1.x(\lambda u_2.yu_2)u_1)z\ \}.$$

4 Inhabitation

In this section we present some applications of the formula-tree proof method in the area of inhabitation of simple types.

Based on Conditions i–iii in Subsection 3.3 one can easily define a proof-tree search algorithm. Note that this can also be seen as an algorithm for searching normal inhabitants, due to the following facts:

- τ has a normal inhabitant iff there is a proof-tree for τ;
- every proof-tree for τ corresponds to a finite number of normal inhabitants and algorithms have been given to compute these inhabitants;
- every normal inhabitant of τ corresponds to exactly one proof-tree for τ and algorithms have been given to compute this proof-tree.

We sketch the main lines of a search-algorithm for proof-trees of a type τ.
Begin the construction of a proof-tree with $p_{\langle\rangle}$, the unique primitive part of form (P1) in FT_τ.
Now, try to complete the proof-tree. For this insert a primitive part with the matching head-variable at the tail-variable of $p_{\langle\rangle}$. Proceed as long as there are primitive parts with tail-variables in the tree and without descendants at these tail-variables, subject to the stopping conditions below.
During the construction of subtrees only use primitive parts that are allowed by Condition ii. Call these the available parts at a tail-variable in an occurrence of a primitive-part in the proof-tree.
Explore all possibilities, using all available primitive parts with the matching head-variable.
The search stops in one of two ways. With success, if a proof-tree is obtained. With failure, whenever trying to proceed at some tail-variable which is the same type-variable and with the same set of available primitive parts as in one of the steps before. This means that there is no need to construct proof-trees with height greater than the number of different type-variables times the number of primitive parts. Eventually, one of the two halting conditions will be obtained, since the number of atoms and primitive parts found in FT_τ is finite.

4.1 Proof-trees in the $\mathbf{BCK}$- *and* $\mathbf{BCI}$-*calculus*

Given a type τ and a proof-tree PT for τ, recall that we denote by $\mathbf{t}_{\mathtt{PT}}$ the tree labelled with nodes of FT_τ obtained from PT and by $N_{\mathbf{t}_{\mathtt{PT}}}$ the standard long normal inhabitant of τ with Böhm-tree $\mathtt{BT}(\mathbf{t}_{\mathtt{PT}})$. Examining algorithm $\mathtt{BT}(\_)$ described in the proof of Theorem 3.18 as well as Definition 3.21, it becomes obvious that

1. $\mathtt{BT}(\mathbf{t}_{\mathtt{PT}})$ is isomorphic to the subtree of PT rooted in the node below its top-node labelled by $p_{\langle\rangle}$;
2. if a node in this subtree of PT is labelled by some primitive part p_ι and its ancestor (in PT) is the kth tail-variable of some primitive part $p_{\iota'}$, then the corresponding node in $\mathtt{BT}(\mathbf{t}_{\mathtt{PT}})$ is of the form $\lambda x_{\iota_1} \ldots x_{\iota_n}.x_\iota$, where $p_{\iota_1}, \ldots p_{\iota_n}$ are the direct descendants of the kth tail-variable of $p_{\iota'}$ in FT_τ.

This observation allows us to identify the following conditions on a proof-tree in order to give origin to terms in the different subsystems of λ-calculus.

A λ-term M is a $\mathbf{BCK}$-term iff every variable occurs at most once in it. Since the class of $\mathbf{BCK}$-term is closed under η-reduction as well as η-expansion, all $\mathbf{BCK}$-inhabitants of a type can be obtained from its long $\mathbf{BCK}$-inhabitants. Furthermore, given a long $\mathbf{BCK}$-inhabitant M it is easy to see that the corresponding standard term $f_s(M)$ is also a $\mathbf{BCK}$-term. Finally, every standard long normal $\mathbf{BCK}$-inhabitant represents only one long normal inhabitant (itself). We conclude from items 1 and 2 above that a proof-tree PT represents a long standard $\mathbf{BCK}$-term with Böhm-tree $\mathtt{BT}(\mathbf{t}_{\mathtt{PT}})$ iff every primitive part in FT_{ι} occurs at most once in PT.

A λ-term M is a $\mathbf{BCI}$-term iff every variable occurs exactly once in it. Using a similar argument as for $\mathbf{BCK}$-terms one concludes that a proof-tree PT represents a $\mathbf{BCI}$-term iff every primitive part in FT_{ι} occurs exactly once in PT.

The previous characterization provides us with an easy justification for the fact that any type τ has at most a finite number of $\mathbf{BCK}$- (and in particular $\mathbf{BCI}$-)inhabitants. In fact, since FT_{τ} consists of a finite number of primitive parts, there exists at most a finite number of proof-trees for τ in which no primitive part occurs more than (exactly) once. Finally, any such proof-tree corresponds to a finite number of normal inhabitants.

4.2 Principal proof-trees

Consider a type τ and a proof-tree PT for τ. We know from Proposition 3.12 that either all or none of the long normal inhabitants in the finite expansion-set $\mathcal{C}(N_{\mathbf{t}_{\mathtt{PT}}})$ are principal inhabitants of $N_{\mathbf{t}_{\mathtt{PT}}}$. If the former is the case, then we call PT a *principal proof-tree*. The purpose of this subsection is to give a syntactic characterization of principal proof-trees (this will allow us to generate principal proof-trees by generating proof-trees which match the conditions given by this syntactic characterization). Note that, if a long normal inhabitant is not a principal inhabitant of τ, then neither are any of the members in its finite η-family. This follows from the fact that principality is preserved by η-expansion. On the other hand, this is not the case for η-reduction and consequently one cannot guarantee that all members of the η-family of a principal long normal inhabitant are also principal. This can be summarized as follows.

- τ has a principal normal inhabitant iff it has a principal proof-tree.

- In order to find all principal normal inhabitants of τ, we have to find all principal proof-trees PT of τ, compute the corresponding standard long normal terms $N_{\mathbf{t}_{\mathtt{PT}}}$, compute the η-families of the members in their expansion-sets $\mathcal{C}(N_{\mathtt{PT}})$ and test the terms in these η-families for principality.

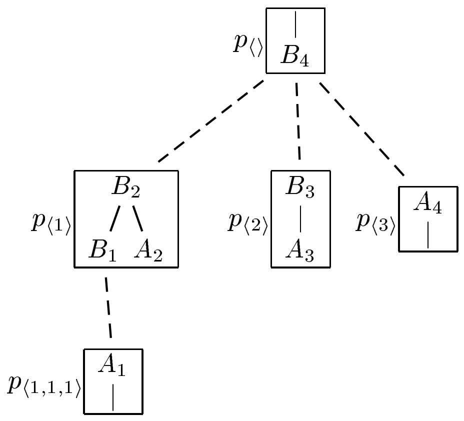
In the following we present a necessary and sufficient condition on PT in order to be a principal proof-tree for τ.

DEFINITION 4.1
The indexed counterpart $\tau_{\mathtt{i}}$ of a type τ is obtained by successively indexing all occurrences of type-variables in τ.

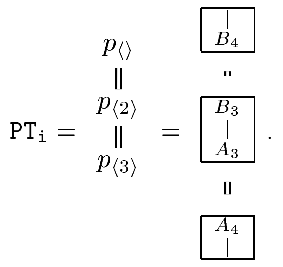
EXAMPLE 4.2
The indexed counterpart $\tau_{\mathtt{i}}$ of τ from Example 3.23 is

$$\tau_{\mathtt{i}} = ((A_1 \to B_1) \to A_2 \to B_2) \to (A_3 \to B_3) \to A_4 \to B_4$$

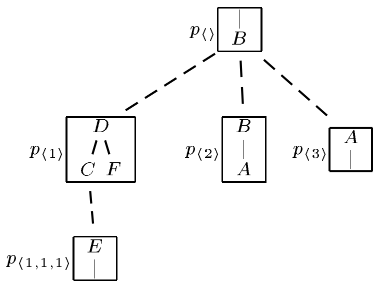
and has the following formula-tree.



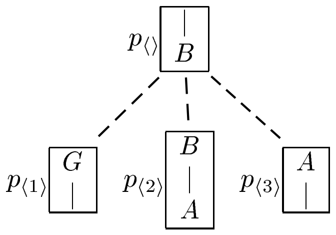
Now consider (the indexed version of) the first proof-tree from Example 3.23:



A proof-tree like this one defines a binary relation between occurences of each type-variable, namely in this case $B_4 = B_3$ and $A_3 = A_4$. If we consider the equivalence relation defined by the reflexive, symmetric and transitive closure of this relation and if for one variable A one has more than one equivalence class, then we can replace all the occurrences of each class by a new variable and the tree is still a proof-tree for this, more general, type. So, in order for the tree to be principal it must cause all occurrences of a variable to be equivalent. In the sequel we use $Eq(\mathtt{PT_i})$ to denote the binary relation defined by $\mathtt{PT_i}$. Here, $Eq(\mathtt{PT_i}) = \{B_4 = B_3, A_3 = A_4\}$ and we conclude that the (unique) long normal inhabitant corresponding to this proof-tree, and which is $\lambda xyz.yz$, is no principal inhabitant of τ. In fact, $Eq(\mathtt{PT_i})$ shows us that there is no need for B_1 and B_2 to correspond to occurrences of the same type-variable and we can replace them respectively by new variables C and D. Similarly, we can replace A_1 and A_2 with E and F, obtaining the more general (but still not principal) type $((E \to C) \to F \to D) \to (A \to B) \to A \to B$ for $\lambda xyz.yz$. Clearly, $\mathtt{PT}$ is also a proof-tree for this formula with formula-tree



Furthermore, since there is no occurrence of primitive part $p_{\langle 1\rangle}$ in PT, this would still be a proof-tree for a formula in which the subtree rooted in position $\langle 1\rangle$ would be as simple as possible. In fact, PT is still a proof-tree for $G \to (A \to B) \to A \to B$ (the principal type of $\lambda xyz.yz$) with formula-tree



The example above shows that in order for a proof-tree to be principal it must use all the primitive parts of form (P2). The previous discussion suggests the following simple characterization of principal proof-trees.

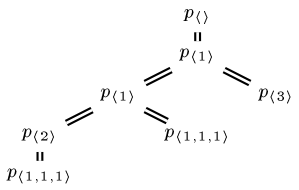
PROPOSITION 4.3
Let τ be a simple type, PT a proof-tree for τ and consider the corresponding proof-tree $\mathtt{PT}_{\mathtt{i}}$ obtained from the indexed counterpart $\tau_{\mathtt{i}}$. Then, PT is a principal proof-tree for τ iff

- every primitive part of form (P2) in FT_{τ} occurs at least once in PT;
- $Eq(\mathtt{PT}_{\mathtt{i}})$ identifies all occurrences of same type-variables in $\tau_{\mathtt{i}}$.

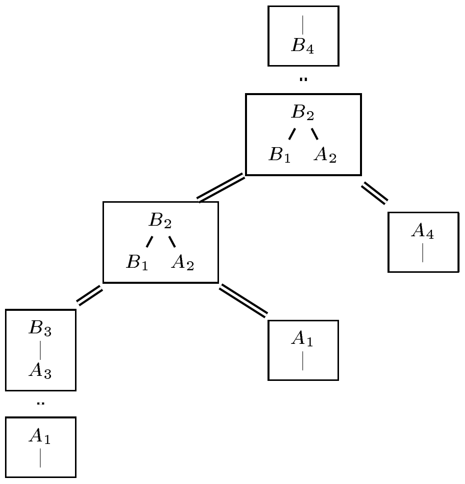
PROOF. We begin showing that if one of the conditions above is not satisfied, then there is another type φ such that τ is an instance of φ and PT is still a proof-tree for φ, i.e. satisfies Conditions i–iii before Proposition 3.22 w.r.t. the formula-tree FT_{φ}. In fact, if one primitive part p_{ι} of form (P2) does not occur in PT, then also none of its descendants does (cf. Condition iii). Thus, if we replace in FT_{τ} the whole subtree rooted in position ι by a primitive part of the form $\boxed{\begin{smallmatrix} C \\ | \end{smallmatrix}}$, where C is a new type-variable, then PT is still a proof-tree for this new formula-tree, which is the formula-tree of the formula φ obtained from τ by replacing the whole negative subpremiss at position ι in φ by C. Note that τ is clearly an instance of φ. On the other hand, consider the equivalence relation on occurrences of type-variables determined by $Eq(\mathtt{PT}_{\mathtt{i}})$ and suppose that there is a type-variable A such that there is more than one equivalence class containing occurrences of A. Consider one of these equivalence classes $\{A_{k_1}, \ldots, A_{k_m}\}$, $m \geq 1$, and consider the formula φ obtained from τ by substituting the occurrences corresponding to $A_{k_1}, \ldots, A_{k_m}$ by a new variable C and maintaining the name A for all other occurrences of A. Again τ is an instance of φ and PT a proof-tree for φ.

Now, suppose that both conditions above are satisfied by PT. Since $Eq(\mathtt{PT}_\mathtt{i})$ identifies all occurrences of type-variables, this means that whenever a type-variable A has more than one occurrence in τ, then all primitive parts with occurrences of A have to appear in PT. Also if we consider any type φ obtained from τ by replacing one or more of these occurrences by a new type-variable, then PT cannot be a proof-tree for φ and consequently φ is not a type of the standard term $N_{\mathbf{t}_{\mathtt{PT}}}$. On the other hand, since all primitive parts of form (P2) occur in PT and considering the remarks made on variables with more than one occurrence in τ we conclude that the only primitive parts in FT_τ that may not occur in PT are of form $\boxed{\begin{smallmatrix} C \\ \mid \end{smallmatrix}}$ where C has exactly one (negative) occurrence in τ. So a more general type with proof-tree PT can only be obtained from τ by deletion of some of these negative occurrences of variables. But, then $N_{\mathbf{t}_{\mathtt{PT}}}$ is no longer an inhabitant of this type due to its definition in the proof of Theorem 3.18. ∎

EXAMPLE 4.4
Consider the last proof-tree PT from Example 3.23.



whose indexed version is



For this proof-tree we have

$$Eq(\mathtt{PT}_\mathtt{i}) = \{B_4 = B_2, B_1 = B_2, A_2 = A_4, B_1 = B_3, A_2 = A_1, A_3 = A_1\}$$

which implies $A_1 = A_2 = A_3 = A_4$ as well as $B_1 = B_2 = B_3 = B_4$. Also, all primitive parts of form (P2) have occurrences in PT. We conclude that τ is the most simple, thus the principal, type for $N_{\mathtt{PT}}$ and that consequently all members of the corresponding set of long normal inhabitants

$$\mathcal{C}(N_{\mathtt{PT}}) =_\alpha \{\lambda xyz.x(\lambda u_1.x(\lambda u_2.yu_1)u_1)z,\ \lambda xyz.x(\lambda u_1.x(\lambda u_2.yu_2)u_1)z\ \}$$

are also principal inhabitants of τ. The only other candidates for principal inhabitants and corresponding to this proof-tree are the remaining members of their η-families, i.e. the terms
$\lambda xy.x(\lambda u_1.x(\lambda u_2.yu_1)u_1)$, $\lambda xy.x(\lambda u_1.x(\lambda u_2.yu_2)u_1)$, $\lambda xy.x(x(\lambda u_2.yu_2))$,
$\lambda xy.x(\lambda u_1.xyu_1)$, $\lambda xy.x(xy)$, $\lambda xyz.x(x(\lambda u_2.yu_2))z$, $\lambda xyz.x(xy)z$,
$\lambda xyz.x(\lambda u_1.xyu_1)z$ and $\lambda xyz.x(xy)z$. None of these terms is a principal inhabitant of τ. In order to obtain all normal principal inhabitants we should consider all possible proof-trees for τ (in this case an infinite number).

4.3 *A context-free grammar generating standard long normal inhabitants*

In [22] it was shown that it is possible to describe the set of normal inhabitants of a type, using an infinitary extension of the concept of context-free grammar, which allows for an infinite number of non-terminal symbols as well as production rules. The set of normal inhabitants of τ corresponds then to the set of terms generated by this, possibly infinitary, grammar plus all terms obtained from those by η-reduction. Also a description of the set of long normal inhabitants in the system with the total discharge convention has been given in [22]. The results in Subsection 2.4 show now, that one can in fact obtain from this grammar all long normal inhabitants, using for this algorithm $\mathcal{C}(\_)$.

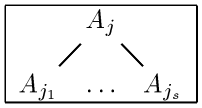
In this subsection we will, for every type τ, define a context-free grammar $\mathtt{G}_\tau$ which generates the set of its standard long normal inhabitants.

DEFINITION 4.5
Let τ be a type, with atoms $A_1, \dots, A_m$ and such that FT_τ has primitive parts $p_{\langle\rangle}, p_{\iota_1}, \dots, p_{\iota_n}$. Let $\mathtt{G}_\tau = (T, N, R, S)$ be the context-free grammar with

- set of terminal symbols $T = \{(,), \lambda, ., x_{\iota_1}, \dots, x_{\iota_n}\}$,
- set of non-terminal symbols $N = \{S\} \cup \{A_i^P \mid 1 \leq i \leq m, P \in 2^{\{\iota_1,\dots,\iota_n\}}\}$
- start symbol S

and such that R is the smallest set satisfying the following conditions.

- If $p_{\langle\rangle} = \boxed{\begin{array}{c} | \\ A_s \end{array}}$ has direct descendants $p_{\langle 1\rangle}, \dots, p_{\langle t\rangle}$ in FT_τ where $t \geq 1$, then R has exactly one production rule for S which is
$$S \to \lambda x_{\langle 1\rangle} \dots x_{\langle t\rangle}.A_s^{\{\langle 1\rangle,\dots,\langle t\rangle\}}$$
- whenever a non-terminal symbol A_j^P appears on the right side of a production rule in R, then for every $\iota \in P$ such that p_ι is of the form $\boxed{\begin{array}{c} A_j \\ | \end{array}}$ there is a rule in R of the form $A_j^P \to x_\iota$
- whenever a non-terminal symbol A_j^P appears on the right side of a production rule in R, then for every $\iota \in P$ such that p_ι is of the form



and such that in FT_τ each A_{j_l} has direct descendants $p_{\iota_l^1}, \dots, p_{\iota_l^{t_l}}$, there is a rule in R of the form
$$A_j^P \to x_\iota(\lambda x_{\iota_1^1} \dots x_{\iota_1^{t_1}}.A_{j_1}^{P_1}) \dots (\lambda x_{\iota_s^1} \dots x_{\iota_s^{t_s}}.A_{j_s}^{P_s})$$
where $P_l = P \cup \{\iota_l^1, \dots, \iota_l^{t_l}\}$, for $1 \leq l \leq s$.

Note that in the previous definition an A^P represents a non-terminal symbol that produces terms of type A. From now on we will usually abbreviate the representation of sets of sequences in superscripts, writing $A^{1;1,2,3;1,2,1}$ instead of $A^{\{\langle 1\rangle,\langle 1,2,3\rangle,\langle 1,2,1\rangle\}}$.

EXAMPLE 4.6
For τ as in Example 3.23, there is $\mathtt{G}_\tau = (T, N, R, S)$ with

- $T = \{(,), \lambda, ., x_{\langle 1\rangle}, x_{\langle 2\rangle}, x_{\langle 3\rangle}, x_{\langle 1,1,1\rangle}\}$,
- $$\begin{array}{rl} N = & \{S\} \cup \\ & \{A^P \mid P \subseteq \{\langle 1\rangle, \langle 2\rangle, \langle 3\rangle, \langle 1,1,1\rangle\}\} \cup \\ & \{B^P \mid P \subseteq \{\langle 1\rangle, \langle 2\rangle, \langle 3\rangle, \langle 1,1,1\rangle\}\} \end{array},$$
- and production rules

$$\begin{array}{lcl}
S & \to & \lambda x_{\langle 1\rangle} x_{\langle 2\rangle} x_{\langle 3\rangle} .\ B^{1;2;3} \\
B^{1;2;3} & \to & x_{\langle 1\rangle}(\lambda x_{\langle 1,1,1\rangle} .\ B^{1;2;3;1,1,1})(\ A^{1;2;3}) \\
B^{1;2;3} & \to & x_{\langle 2\rangle}(\ A^{1;2;3}) \\
B^{1;2;3;1,1,1} & \to & x_{\langle 1\rangle}(\lambda x_{\langle 1,1,1\rangle} .\ B^{1;2;3;1,1,1})(\ A^{1;2;3;1,1,1}) \\
B^{1;2;3;1,1,1} & \to & x_{\langle 2\rangle}(\ A^{1;2;3;1,1,1}) \\
A^{1;2;3} & \to & x_{\langle 3\rangle} \\
A^{1;2;3;1,1,1} & \to & x_{\langle 3\rangle} \\
A^{1;2;3;1,1,1} & \to & x_{\langle 1,1,1\rangle}
\end{array}$$

This grammar generates the following infinite set of standard long normal inhabitants of τ.

$$\begin{array}{rl}
\mathcal{L}(\mathtt{G}_\tau) = \{ & \lambda x_{\langle 1\rangle} x_{\langle 2\rangle} x_{\langle 3\rangle} . x_{\langle 2\rangle} x_{\langle 3\rangle}, \\
& \lambda x_{\langle 1\rangle} x_{\langle 2\rangle} x_{\langle 3\rangle} . x_{\langle 1\rangle}(\lambda x_{\langle 1,1,1\rangle} . x_{\langle 2\rangle} x_{\langle 3\rangle}) x_{\langle 3\rangle}, \\
& \lambda x_{\langle 1\rangle} x_{\langle 2\rangle} x_{\langle 3\rangle} . x_{\langle 1\rangle}(\lambda x_{\langle 1,1,1\rangle} . x_{\langle 2\rangle} x_{\langle 1,1,1\rangle}) x_{\langle 3\rangle}, \\
& \lambda x_{\langle 1\rangle} x_{\langle 2\rangle} x_{\langle 3\rangle} . x_{\langle 1\rangle}(\lambda x_{\langle 1,1,1\rangle} . x_{\langle 1\rangle}(\lambda x_{\langle 1,1,1\rangle} . x_{\langle 2\rangle} x_{\langle 3\rangle}) x_{\langle 3\rangle}) x_{\langle 3\rangle}, \\
& \lambda x_{\langle 1\rangle} x_{\langle 2\rangle} x_{\langle 3\rangle} . x_{\langle 1\rangle}(\lambda x_{\langle 1,1,1\rangle} . x_{\langle 1\rangle}(\lambda x_{\langle 1,1,1\rangle} . x_{\langle 2\rangle} x_{\langle 3\rangle}) x_{\langle 1,1,1\rangle}) x_{\langle 3\rangle}, \\
& \lambda x_{\langle 1\rangle} x_{\langle 2\rangle} x_{\langle 3\rangle} . x_{\langle 1\rangle}(\lambda x_{\langle 1,1,1\rangle} . x_{\langle 1\rangle}(\lambda x_{\langle 1,1,1\rangle} . x_{\langle 2\rangle} x_{\langle 1,1,1\rangle}) x_{\langle 3\rangle}) x_{\langle 3\rangle}, \\
& \lambda x_{\langle 1\rangle} x_{\langle 2\rangle} x_{\langle 3\rangle} . x_{\langle 1\rangle}(\lambda x_{\langle 1,1,1\rangle} . x_{\langle 1\rangle}(\lambda x_{\langle 1,1,1\rangle} . x_{\langle 2\rangle} x_{\langle 1,1,1\rangle}) x_{\langle 1,1,1\rangle}) x_{\langle 3\rangle}, \ldots \}
\end{array}$$

Thus,

$$\begin{array}{rll}
\mathtt{Nhabs}(\tau) & = \{ & M \mid \exists\ T \in \mathcal{L}\ \exists N \in \mathcal{C}(T) \text{ such that } N \to_\eta M\} \\
& =_\alpha \{ & \lambda xyz.yz, \\
&& \lambda xy.y, \\
&& \lambda xyz.x(\lambda u_1.yz)z, \\
&& \lambda xuz.x(\lambda u_1.yu_1)z, \\
&& \lambda xy.x(\lambda u_1.yu_1, \\
&& \lambda xyz.xyz, \\
&& \lambda xy.xy, \\
&& \lambda xyz.x(\lambda u_1.x(\lambda u_2.yz)z)z, \\
&& \lambda xyz.x(\lambda u_1.x(\lambda u_2.yz)u_1)z, \\
&& \lambda xyz.x(x(\lambda u_2.yz))z, \\
&& \lambda xyz.x(\lambda u_1.x(\lambda u_2.yu_1)z)z, \\
&& \lambda xyz.x(\lambda u_1.x(\lambda u_2.yu_2)z)z, \\
&& \lambda xyz.x(\lambda u_1.xyz)z, \ldots \}
\end{array}$$

PROPOSITION 4.7
For any type τ the grammar $\mathtt{G}_\tau$ describes $\mathtt{Nhabs}(\tau)$ in the sense that

$$\mathtt{Nhabs}(\tau) = \{M \mid \exists T \in \mathcal{L}(\mathtt{G}_\tau)\ \exists N \in \mathcal{C}(T) \text{ such that } N \to_\eta M\}.$$

PROOF. It is sufficient to show that $\mathtt{G}_\tau$ generates the set of standard long normal inhabitants of τ. The intuitive meaning of the start-symbol S is a standard long normal inhabitant of τ and each non-terminal symbol $A_i^{\{\iota_1,\dots,\iota_n\}}$ represents a subterm of the form $x_\iota M_1 \dots M_k$, where $k \geq 0$ and $\iota \in \{\iota_1,\dots,\iota_n\}$, of atomic type A_i and in the scope of variables $x_{\iota_1},\dots,x_{\iota_n}$, respectively with types $\widetilde{(\iota_1)}_\tau,\dots,\widetilde{(\iota_n)}_\tau$. In the following we show that this interpretation is consistent with the definition of $\mathtt{G}_\tau$ and that $\mathtt{G}_\tau$ generates all existing standard long normal inhabitants of τ.

Unless $\tau = A_1$ and has no inhabitant, there is $\tau = \tau_1 \to \dots \to \tau_t \to A_s$, with $1 \leq s \leq m$ and $t \geq 1$. Then, $p_{\langle\rangle} = \boxed{\begin{array}{c} | \\ A_s \end{array}}$ has direct descendants $p_{\langle 1\rangle},\dots,p_{\langle t\rangle}$ in FT_τ. Due to the structure of τ every standard long normal inhabitant S has to be of the form $\lambda x_{\langle 1\rangle}\dots x_{\langle t\rangle}.A_s^{\{\langle 1\rangle,\dots,\langle t\rangle\}}$, where $A_s^{\{\langle 1\rangle,\dots,\langle t\rangle\}}$ is a term of the form $x_\iota M_1 \dots M_k$, $k \geq 0$ and $\iota \in \{\langle 1\rangle,\dots,\langle t\rangle\}$, of atomic type $A_s = \mathtt{tail}(\tau)$ and in the scope of variables $x_{\langle 1\rangle},\dots,x_{\langle n\rangle}$, respectively with types $\widetilde{\langle 1\rangle}_\tau = \tau_1,\dots,\widetilde{\langle t\rangle}_\tau = \tau_t$.

Now, suppose we want to construct a subterm A_j^P of the form $x_\iota M_1 \dots M_k$, where $k \geq 0$ and $\iota \in \{\iota_1,\dots,\iota_n\}$, of atomic type A_i and in the scope of the variables indexed by the sequences in P. If there is $\iota \in P$ such that p_ι is of the form $\boxed{\begin{array}{c} A_j \\ | \end{array}}$, then $\widetilde{\iota}_\tau = A_j$ is the type of x_ι and $A_j^P = x_\iota$ is a possible choice. If there is $\iota \in P$ such that p_ι is of the form

$$\boxed{\begin{array}{ccc} & A_j & \\ \diagup & & \diagdown \\ A_{j_1} & \dots & A_{j_s} \end{array}}$$

and such that in FT_τ each A_{j_l} has direct descendants $p_{\iota_l^1},\dots,p_{\iota_l^{t_l}}$, then x_ι has type $\widetilde{\iota}_\tau = (\widetilde{\iota_1^1}_\tau \to \dots \widetilde{\iota_1^{t_1}}_\tau \to A_{j_1}) \to \dots \to (\widetilde{\iota_s^1}_\tau \to \dots \widetilde{\iota_s^{t_s}}_\tau \to A_{j_s}) \to A_j$. Thus, $A_j^P = x_\iota(\lambda x_{\iota_1^1}\dots x_{\iota_1^{t_1}}.A_{j_1}^{P_1})\dots(\lambda x_{\iota_s^1}\dots x_{\iota_s^{t_s}}.A_{j_s}^{P_s})$, where $P_l = P \cup \{\iota_l^1,\dots,\iota_l^{t_l}\}$, for $1 \leq l \leq s$, is a possible choice. ∎

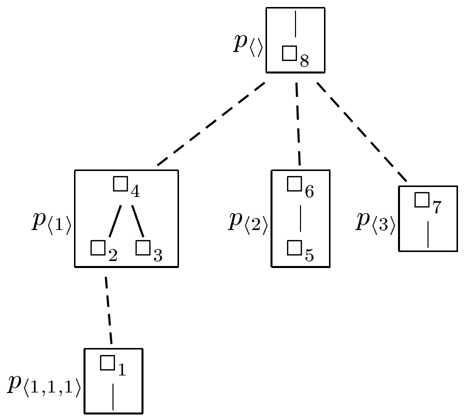
4.4 A common grammar scheme for types with the same structure

In this subsection we show that the normal inhabitants of types with the same structure can be generated by identical context-free grammars, up to renaming of symbols. Given a type τ we represent the corresponding type-structure replacing occurrences of atoms by blank successively indexed boxes as illustrated in the following example.

EXAMPLE 4.8
The type-structure corresponding to τ from Example 3.23 is

$$\Theta = ((\square_1 \to \square_2) \to \square_3 \to \square_4) \to (\square_5 \to \square_6) \to \square_7 \to \square_8$$

with formula tree



Possible instances, i.e types with this structure, are for example,

1. for $\square_1 = \square_2 = \square_3 = \square_5 = \square_7 = A$ and $\square_4 = \square_6 = \square_8 = B$ the type
$$\tau_1 = ((A \to A) \to A \to B) \to (A \to B) \to A \to B$$
2. for $\square_1 = \square_2 = A$, $\square_3 = \square_7 = B$ and $\square_4 = \square_5 = \square_6 = \square_8 = C$ the type
$$\tau_2 = ((A \to A) \to B \to C) \to (C \to C) \to B \to C$$
3. or for $\square_1 = \square_2 = \square_3 = \square_4 = \square_5 = \square_6 = \square_8 = A$ and $\square_7 = B$ the type
$$\tau_3 = ((A \to A) \to A \to A) \to (A \to A) \to B \to A.$$

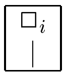
In the following we define an algorithm which given a type-structure Θ computes a grammar(-scheme) $\mathtt{G}_\Theta$ such that for every instance τ with structure Θ a context-free grammar equivalent to $\mathtt{G}_\tau$ can be obtained from $\mathtt{G}_\Theta$ by substitution of all occurrences of indexed boxes in this grammar by the corresponding atoms in τ.

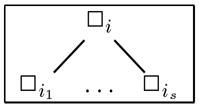
DEFINITION 4.9
Let Θ be a type-structure, with indexed boxes $\square_1, \ldots, \square_m$ and such that FT_Θ has primitive parts $p_{\langle\rangle}, p_{\iota_1}, \ldots, p_{\iota_n}$. Let $\mathtt{G}_\Theta = (T, N, R, S)$ be the context-free grammar with

- set of terminal symbols $T = \{(,), \lambda, ., x_{\iota_1}, \ldots, x_{\iota_n}\}$,
- set of non-terminal symbols $N = \{S\} \cup \{\square_i^P | 1 \leq i \leq m, P \in 2^{\{\iota_1,\ldots,\iota_n\}}\}$,
- start symbol S,

and such that R is the smallest set satisfying the following conditions.

- If $p_{\langle\rangle} = \boxed{\begin{array}{c} | \\ \square_n \end{array}}$ has direct descendants $p_{\langle 1\rangle}, \ldots, p_{\langle t\rangle}$ in FT_τ where $t \geq 1$, then R has exactly one production rule for S which is
$$S \to \lambda x_{\langle 1\rangle} \ldots x_{\langle t\rangle}.\square_n^{\{\langle 1\rangle,\ldots,\langle t\rangle\}}$$

- whenever a non-terminal symbol $\Box_j^P$ appears on the right side of a production rule in R, then for every $\iota \in P$ such that p_ι is of the form  there is a rule in R of the form $\Box_i^P \to x_\iota$
- whenever a non-terminal symbol $\Box_j^P$ appears on the right side of a production rule in R, then for every $\iota \in P$ such that p_ι is of the form



and such that in FT_τ each $\Box_{i_l}$ has direct descendants $p_{\iota_l^1}, \dots, p_{\iota_l^{t_l}}$, there is a rule in R of the form

$$\Box_i^P \to x_\iota(\lambda x_{\iota_1^1} \dots x_{\iota_1^{t_1}}.\Box_{i_1}^{P_1}) \dots (\lambda x_{\iota_s^1} \dots x_{\iota_s^{t_s}}.\Box_{i_s}^{P_s})$$

where $P_l = P \cup \{\iota_l^1, \dots, \iota_l^{t_l}\}$, for $1 \le l \le s$.

EXAMPLE 4.10
Let Θ be as in Example 4.8. Then, $\mathtt{G}_\Theta = (T, N, R, S)$ with

- $T = \{(,), \lambda, ., x_{\langle 1\rangle}, x_{\langle 2\rangle}, x_{\langle 3\rangle}, x_{\langle 1,1,1\rangle}\}$,
- $N = \{S\} \cup \{\Box_i^P \mid 1 \le i \le 8, P \in 2^{\{\langle 1\rangle,\langle 2\rangle,\langle 3\rangle,\langle 1,1,1\rangle\}}\}$,
- start symbol S,

and R given by

$$\begin{array}{lcl}
S & \to & \lambda x_{\langle 1\rangle} x_{\langle 2\rangle} x_{\langle 3\rangle}.\, \Box_8^{1;2;3} \\
\Box_4^{1;2;3} & \to & x_{\langle 1\rangle}(\lambda x_{\langle 1,1,1\rangle}.\, \Box_2^{1;2;3;1,1,1})(\, \Box_3^{1;2;3}) \\
\Box_6^{1;2;3} & \to & x_{\langle 2\rangle}(\, \Box_5^{1;2;3}) \\
\Box_7^{1;2;3} & \to & x_{\langle 3\rangle} \\
\Box_1^{1;2;3;1,1,1} & \to & x_{\langle 1,1,1\rangle} \\
\Box_4^{1;2;3;1,1,1} & \to & x_{\langle 1\rangle}(\lambda x_{\langle 1,1,1\rangle}.\, \Box_2^{1;2;3;1,1,1})(\, \Box_3^{1;2;3;1,1,1}) \\
\Box_6^{1;2;3;1,1,1} & \to & x_{\langle 2\rangle}(\, \Box_5^{1;2;3;1,1,1}) \\
\Box_7^{1;2;3;1,1,1} & \to & x_{\langle 3\rangle}.
\end{array}$$

Naturally, for every type-scheme Θ the context-free grammar $\mathtt{G}_\Theta$ generates the empty language, since no non-terminal symbol on the right side of production rules will appear on the left side of any rule in R. But, the following proposition states that $\mathtt{G}_\Theta$ can in fact be seen as a scheme representing grammars for any type with structure Θ.

PROPOSITION 4.11
Let τ be a type with structure Θ, i.e. $\tau = \Theta[A_1/\Box_1, \dots, A_n/\Box_n]$, for, not necessarily distinct, atoms $A_1, \dots, A_n$. Let $\mathtt{G}_{\tau/\Theta}$ be the context-free grammar obtained from $\mathtt{G}_\Theta$ by substituting all occurrences of $\Box_1, \dots, \Box_n$ respectively by $A_1, \dots, A_n$. Then (modulo α-conversion),

$$\begin{array}{ll}
\mathtt{Nhabs}(\tau) & = \{M \mid \exists T \in \mathcal{L}(\mathtt{G}_\tau)\ \exists N \in \mathcal{C}(T) \text{ such that } N \to_\eta M\} \\
& = \{M \mid \exists T \in \mathcal{L}(\mathtt{G}_{\tau/\Theta})\ \exists N \in \mathcal{C}(T) \text{ such that } N \to_\eta M\}.
\end{array}$$

PROOF. It is sufficient to verify that $\mathcal{L}(\mathtt{G}_\tau) = \mathcal{L}(\mathtt{G}_\Theta)$. For this, note first that every production rule of $\mathtt{G}_\tau$ is a production rule of $\mathtt{G}_{\tau/\Theta}$. On the other hand, no other rule in $\mathtt{G}_{\tau/\Theta}$ will ever be used during the generation of a term-scheme. ∎

EXAMPLE 4.12
For the types τ_1, τ_2 and τ_3 in Example 4.8 we obtain the following results.

1. $\mathtt{G}_{\tau_1/\Theta}$ is obtained from $\mathtt{G}_\Theta$ substituting all occurrences of $\square_1, \square_2, \square_3, \square_5$ and $\square_7$ by the atom A and all occurrences of $\square_4, \square_6$ and $\square_8$ by B. The resulting set R has the following production rules

$$\begin{array}{lll} S & \rightarrow & \lambda x_{\langle 1\rangle} x_{\langle 2\rangle} x_{\langle 3\rangle}.B^{1;2;3} \\ B^{1;2;3} & \rightarrow & x_{\langle 1\rangle}(\lambda x_{\langle 1,1,1\rangle}.A^{1;2;3;1,1,1})(A^{1;2;3}) \\ B^{1;2;3} & \rightarrow & x_{\langle 2\rangle}(A^{1;2;3}) \\ A^{1;2;3} & \rightarrow & x_{\langle 3\rangle} \\ A^{1;2;3;1,1,1} & \rightarrow & x_{\langle 1,1,1\rangle} \\ B^{1;2;3;1,1,1} & \rightarrow & x_{\langle 1\rangle}(\lambda x_{\langle 1,1,1\rangle}.A^{1;2;3;1,1,1})(A^{1;2;3;1,1,1}) \\ B^{1;2;3;1,1,1} & \rightarrow & x_{\langle 2\rangle}(A^{1;2;3;1,1,1}) \\ A^{1;2;3;1,1,1} & \rightarrow & x_{\langle 3\rangle} \end{array}$$

which can be simplified to

$$\begin{array}{ll} S \rightarrow & \lambda x_{\langle 1\rangle} x_{\langle 2\rangle} x_{\langle 3\rangle}.x_{\langle 1\rangle}(\lambda x_{\langle 1,1,1\rangle}.x_{\langle 1,1,1\rangle})x_{\langle 3\rangle} \\ & \mid \lambda x_{\langle 1\rangle} x_{\langle 2\rangle} x_{\langle 3\rangle}.x_{\langle 1\rangle}(\lambda x_{\langle 1,1,1\rangle}.x_{\langle 3\rangle})x_{\langle 3\rangle} \\ & \mid \lambda x_{\langle 1\rangle} x_{\langle 2\rangle} x_{\langle 3\rangle}.x_{\langle 2\rangle} x_{\langle 3\rangle}. \end{array}$$

Thus,

$$\begin{array}{ll} \mathtt{Nhabs}(\tau_1) =_\alpha \{ & \lambda xyz.x(\lambda u.u)z,\ \lambda xy.x(\lambda u.u), \\ & \lambda xyz.x(\lambda u.z)z,\ \lambda xyz.yz,\ \lambda xy.y\ \ \}. \end{array}$$

2. $\mathtt{G}_{\tau_2/\Theta}$ has production rules

$$\begin{array}{lll} S & \rightarrow & \lambda x_{\langle 1\rangle} x_{\langle 2\rangle} x_{\langle 3\rangle}.C^{1;2;3} \\ C^{1;2;3} & \rightarrow & x_{\langle 1\rangle}(\lambda x_{\langle 1,1,1\rangle}.A^{1;2;3;1,1,1})(B^{1;2;3}) \\ C^{1;2;3} & \rightarrow & x_{\langle 2\rangle}(C^{1;2;3}) \\ B^{1;2;3} & \rightarrow & x_{\langle 3\rangle} \\ A^{1;2;3;1,1,1} & \rightarrow & x_{\langle 1,1,1\rangle} \\ C^{1;2;3;1,1,1} & \rightarrow & x_{\langle 1\rangle}(\lambda x_{\langle 1,1,1\rangle}.A^{1;2;3;1,1,1})(B^{1;2;3;1,1,1}) \\ C^{1;2;3;1,1,1} & \rightarrow & x_{\langle 2\rangle}(C^{1;2;3;1,1,1}) \\ B^{1;2;3;1,1,1} & \rightarrow & x_{\langle 3\rangle} \end{array}$$

which can be simplified to

$$\begin{array}{lll} S & \rightarrow & \lambda x_{\langle 1\rangle} x_{\langle 2\rangle} x_{\langle 3\rangle}.C^{1;2;3} \\ C^{1;2;3} & \rightarrow & x_{\langle 1\rangle}(\lambda x_{\langle 1,1,1\rangle}.x_{\langle 1,1,1\rangle})x_{\langle 3\rangle} \mid x_{\langle 2\rangle}\ (C^{1;2;3}). \end{array}$$

Thus,

$$\begin{array}{ll} \mathtt{Nhabs}(\tau_2) =_\alpha \{ & \lambda xyz.x(\lambda u.u)z, \lambda xy.x(\lambda u.u), \\ & \lambda xyz.y(x(\lambda u.u)z), \lambda xyz.y(y(x(\lambda u.u)z)), \ldots\ \}. \end{array}$$

3. $\mathtt{G}_{\tau_3/\Theta}$ has production rules

$$\begin{array}{lll} S & \rightarrow & \lambda x_{\langle 1\rangle} x_{\langle 2\rangle} x_{\langle 3\rangle}.A^{1;2;3} \\ A^{1;2;3} & \rightarrow & x_{\langle 1\rangle}(\lambda x_{\langle 1,1,1\rangle}.A^{1;2;3;1,1,1})(A^{1;2;3}) \\ A^{1;2;3} & \rightarrow & x_{\langle 2\rangle}(A^{1;2;3}) \\ B^{1;2;3} & \rightarrow & x_{\langle 3\rangle} \\ A^{1;2;3;1,1,1} & \rightarrow & x_{\langle 1,1,1\rangle} \\ A^{1;2;3;1,1,1} & \rightarrow & x_{\langle 1\rangle}(\lambda x_{\langle 1,1,1\rangle}.A^{1;2;3;1,1,1})(A^{1;2;3;1,1,1}) \\ A^{1;2;3;1,1,1} & \rightarrow & x_{\langle 2\rangle}(A^{1;2;3;1,1,1}) \\ B^{1;2;3;1,1,1} & \rightarrow & x_{\langle 3\rangle}. \end{array}$$

This grammar generates the empty language, thus $\mathtt{Nhabs}(\tau_3) = \emptyset$.

5 Studying provability in implicational intuitionistic logic

In this section we illustrate how the formula-tree proof method can be used very efficiently in order to obtain and prove results on provability in (subsystems of) the implicational fragment of intuitionistic logic. As such, we show in Subsection 5.1 that former results concerning uniqueness of $\beta\eta$-normal proofs for certain classes of formulas become straightforward, can in fact be extended and have simple, almost direct proofs. Furthermore, we prove uniqueness of β-normal proofs for a new large class of formulas. In Subsection 5.2 we obtain a simple, syntactical characterization of the set of provable monatomic formulas and obtain as a corollary a necessary condition for intuitionistic theorems in general.

5.1 Uniqueness of normal proofs in implicational intuitionistic logic

In 1986, Komori [19] raised the question whether normal proofs of minimal formulas are unique in the natural deduction system for implicational intuitionistic logic and for $\mathbf{BCK}$-logic. A minimal formula τ is a provable formula such that there is no other provable formula from which τ can be obtained by instantiation (other than by renaming of variables). The latter was shown to be true for β-reduction, and consequently[5] for $\beta\eta$-reduction, by Hirokawa in [16]. On the other hand, minimal implicational intuitionistic theorems with more than one $\beta\eta$-normal proof have been exhibited independently by Mints in [20], who on the other side proved uniqueness of $\beta\eta$-normal proofs for balanced formulas, i.e. for formulas in which no variable occurs more than twice, and by Tatsuta in [23]. In this last paper Tatsuta also showed uniqueness of β-normal proofs for minimal formulas of depth ≤ 2. These results have been somehow unified by Aoto and Ono who showed uniqueness of $\beta\eta$-normal proofs for negatively non-duplicated formulas in [1]. Uniqueness of β-normal proofs for minimal formulas provable without non-prime contraction was shown by Aoto in [2].

In the following we show that using the representation of formulas by formula-trees, the results concerning $\beta\eta$-normal proofs become straightforward, can in fact be extended and have simple, almost direct proofs. Furthermore, we prove uniqueness of β-normal proofs for another large class of formulas.

LEMMA 5.1
Consider a type τ, a proof-tree $\mathtt{PT}$ for τ and the standard long normal inhabitant $N_{\mathbf{t}_{\mathtt{PT}}}$ of τ with Böhm-tree $\mathtt{BT}(\mathbf{t}_{\mathtt{PT}})$. Then, $|\mathcal{C}(N_{\mathbf{t}_{\mathtt{PT}}})| > 1$ only if at least one primitive part has been used more than once in a branch of $\mathtt{PT}$.

PROOF. If no primitive part appears more than once in a branch of $\mathtt{PT}$, then no abstraction sequence appears more than once in any branch of the Böhm-tree $\mathtt{BT}(\mathbf{t}_{\mathtt{PT}})$. This means that in $N_{\mathbf{t}_{\mathtt{PT}}}$ no x_ι occurs in the scope of more than one abstraction λx_ι and consequently $\mathcal{C}(N_{\mathbf{t}_{\mathtt{PT}}})$ has exactly one element. ∎

DEFINITION 5.2
A formula τ is called *negatively non-duplicated* iff every variable has at most one negative occurrence in it.

Note that the class of negatively non-duplicated theorems properly contains most of the classes that are mentioned in the beginning of this subsection, i.e. the class of minimal formulas provable

[5]Note that uniqueness of β-normal proofs implies uniqueness of $\beta\eta$-normal proofs. This results trivially from the fact that every $\beta\eta$-normal form is in particular a β-normal form.

without non-prime contraction, etc. Thus the following result, from [1], implies the results concerning uniqueness of $\beta\eta$-normal proofs also for these classes of formulas. We now show that it becomes almost straightforward when using formula-trees and present a rather simple proof for it.

THEOREM 5.3
Every provable negatively non-duplicated formula has exactly one $\beta\eta$-normal proof.

PROOF. We consider a provable negatively non-duplicated formula τ with formula-tree FT_τ. Remember that head-variables in primitive parts correspond to negative occurrences in τ, while tail-variables correspond to positive occurrences. Since τ is negatively non-duplicated, this means that for each variable A there is at most one primitive part in FT_τ with head-variable A. Thus, in any step during the construction of a proof-tree there is at most one primitive part with the matching head-variable, which justifies the existence of a unique proof-tree. On the other hand, it also shows that there is no possibility of using a primitive part more that once in any branch, since this would lead to an infinite repetition. So the result follows from Lemma 5.1. ∎

The proof of the previous result allows us to identify a larger class of formulas for which uniqueness of $\beta\eta$-reduction is true.

DEFINITION 5.4
A formula τ is called *deterministic* iff in every step during the construction of a proof-tree there is at most one primitive part with the matching head-variable available.

EXAMPLE 5.5
The formula from Example 3.23 is not deterministic, since already in the first step there are two possible choices. In fact, we can choose between primitive parts $p_{\langle 1\rangle}$ or $p_{\langle 2\rangle}$ in order to pursue the construction of a proof-tree.

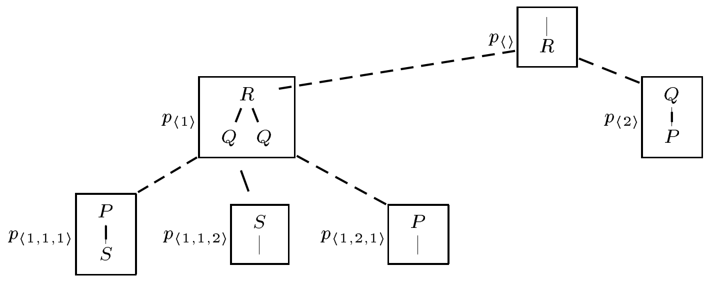
COROLLARY 5.6
Every deterministic formula has exactly one $\beta\eta$-normal proof.

Note that the class of deterministic formulas properly includes the class of negatively non-duplicated formulas, as well as some other classes of formulas for which uniqueness of $\beta\eta$-normal proofs have been shown (cf. Theorem 1.8 in [12] and Theorem 5 in [15]). Furthermore, deterministic formulas are very easily recognized[6].

EXAMPLE 5.7
Consider the following formula τ from [2]. The formula τ is not negatively non-duplicated, hence does not fit into the class of formulas for which uniqueness of $\beta\eta$-reduction has been shown before.

$$\tau = (((S \to P) \to S \to Q) \to (P \to Q) \to R) \to (P \to Q) \to R.$$

Its formula-tree $\texttt{tree}(\tau)$ is



[6]Given a formula τ with n occurrences of variables, it takes at most n steps to decide if τ is deterministic.

and the construction of its unique proof-tree

$$\begin{tikzcd}
 & p_{\langle\rangle} \arrow[d, equal] & \\
 & p_{\langle 1\rangle} \arrow[dl, equal] \arrow[dr, equal] & \\
p_{\langle 2\rangle} \arrow[d, equal] & & p_{\langle 2\rangle} \arrow[d, equal] \\
p_{\langle 1,1,1\rangle} \arrow[d, equal] & & p_{\langle 1,2,1\rangle} \\
p_{\langle 1,1,2\rangle} & &
\end{tikzcd}$$

is completely deterministic. In spite of the fact that there are two positive occurrences of P, namely as head-variable of $p_{\langle 1,1,1\rangle}$ and of $p_{\langle 1,2,1\rangle}$, these two primitive parts are never available at the same time. Since $\langle 1,1,1\rangle = \langle 1\rangle \cdot \langle 1,1\rangle$, $p_{\langle 1,1,1\rangle}$ can only be used in a subtree rooted in the first branch of some occurrence of $p_{\langle 1\rangle}$, while $p_{\langle 1,2,1\rangle}$ can only be used in a subtree rooted in the second branch of an occurrence of $p_{\langle 1\rangle}$, due to $\langle 1,2,1\rangle = \langle 1\rangle \cdot \langle 2,1\rangle$. We conclude that τ has exactly one $\beta\eta$-normal proof.

After analysing negatively non-duplicated formulas it seems just natural to consider formulas with the symmetric property.

DEFINITION 5.8
A formula τ is called *positively non-duplicated* iff every variable has at most one positive occurrence in it.

Since positive occurrences of variables in τ appear as tail-variables in primitive parts in FT_τ, we conclude that τ is positively non-duplicated iff all tail-variables in the primitive parts in FT_τ are distinct. This suggests the following result.

THEOREM 5.9
Every proof of a positively non-duplicated formula τ is a proof in $\mathbf{BCK}$-logic. Furthermore, if τ is provable then it has a finite number of proofs.

PROOF. Consider a positively non-duplicated formula/type τ and a proof-tree `PT` for τ. We conclude from Subsection 4.1 that it is sufficient to show that every primitive part in FT_τ occurs at most once in `PT`. First, note that $p_{\langle\rangle}$ occurs exactly once in `PT` since it has no head-variable. Now, consider an occurrence $\underline{p_\iota}$ of some primitive part p_ι in `PT` descending in `PT` from a tail-variable A of a primitive part $p_{\iota'}$. The head-variable of p_ι is necessarily A. Furthermore, if there is another occurrence of p_ι in `PT`, then also this occurrence has to descend from an occurrence of $p_{\iota'}$, since this is the only primitive part with a tail-variable A. So, also the primitive part above $\underline{p_\iota}$ has more than one occurrence in `PT`. Repeating this argument we ascend in `PT`, concluding eventually that $p_{\langle\rangle}$ has more than one occurrence in `PT`, which is false. The last result follows from the fact that the number of $\mathbf{BCK}$-inhabitants of a type is finite. ∎

In the remaining of this subsection we present a new class of formulas for which we prove uniqueness of β-normal proofs.

DEFINITION 5.10
A formula τ is called η-free iff there is no positive occurrence of a subformula in τ of the form $\alpha_1 \to \ldots \to \alpha_n \to A \to B$, where A and B are variables.

The name η-free is justified by the following.

THEOREM 5.11
All η-families of an η-free type are singletons.

PROOF. If a standard long normal inhabitant M of a type τ is not in η-normal form, then there is some (sub-)term of M of the form

$$\lambda x_{\iota_1} \dots x_{\iota_n} . x M_1 \dots M_{m-1} x_{\iota_n} .$$

Since M is long, x_{ι_n} has to be of atomic type. Thus, the corresponding primitive part p_{ι_n} has to be of the form $\boxed{\begin{array}{c}B\\ |\end{array}}$ for some atom B and $p_{\iota_1}, \dots, p_{\iota_n}$ are all direct descendants of some tail-variable of another primitive part in FT_τ. Consequently, there is some positive occurrence of a subformula in τ of the form $\alpha_1 \to \dots \to \alpha_{n-1} \to B \to A$, where A and B are type-variables. This last conclusion follows from the definition of FT_τ. ∎

COROLLARY 5.12
Every provable negatively non-duplicated and η-free formula has a unique β-normal proof.

COROLLARY 5.13
Every provable deterministic and η-free formula has a unique β-normal proof.

EXAMPLE 5.14
The formula τ from Example 5.7 does not fit into this class, since it has positive occurrences of $(S \to P) \to S \to Q$ as well as $P \to Q$. In fact, there are two β-normal proofs for τ.

5.2 A characterization of monatomic theorems in implicational intuitionistic logic

DEFINITION 5.15
A type τ is called *monatomic* iff only one type variable occurs in τ.

The following result on the cardinality of the set of normal proofs for a monatomic formula was originally given for types and is true for β- as well as for $\beta\eta$-normal forms.

THEOREM 5.16 ([4])
Let τ be a monatomic formula of the form

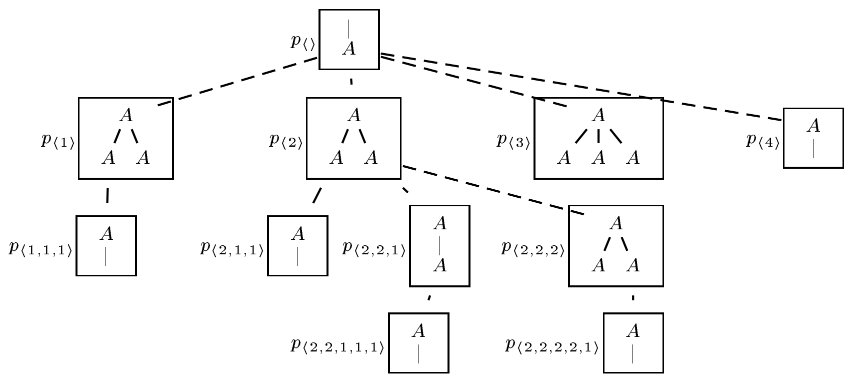
$$\tau = \tau_1 \to \dots \to \tau_m \to A$$

with $m \geq 0$. Then,

(i) if at least one τ_i is composite, i.e. non-atomic, then τ has either none or an infinite number of normal proofs;

(ii) if $\tau_1 = \dots = \tau_m = A$, then τ has exactly m normal proofs.

EXAMPLE 5.17
Consider the formula $\tau = ((A \to A) \to A \to A) \to ((A \to A) \to (((A \to A) \to A) \to (A \to (A \to A) \to A) \to A) \to A) \to (A \to A \to A \to A) \to A \to A$ with formula-tree


$p_{\langle\rangle}$
$p_{\langle 1\rangle}$
$p_{\langle 2\rangle}$
$p_{\langle 3\rangle}$
$p_{\langle 4\rangle}$
$p_{\langle 1,1,1\rangle}$
$p_{\langle 2,1,1\rangle}$
$p_{\langle 2,2,1\rangle}$
$p_{\langle 2,2,2\rangle}$
$p_{\langle 2,2,1,1,1\rangle}$
$p_{\langle 2,2,2,2,1\rangle}$
A

From Ben-Yelles' result [4] we conclude that τ has either none or an infinite number of normal proofs. But does it in fact have any?

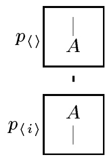
In the following we give a simple, syntactical characterization of the set of provable monatomic formulas.

DEFINITION 5.18
A formula-tree is called *complete* iff all leaves are of the form (P3). *Pruning* a formula-tree means removing primitive parts (together with the primitive parts descending from them).

THEOREM 5.19
A monatomic formula τ is provable iff at least one complete subtree can be obtained by pruning FT_τ.

PROOF. The if part is easy. Given a complete subtree, obtained by pruning FT_τ, just replace dashed lines by doubled lines in this subtree. The result is a proof-tree for τ.

For the only-if part we describe an algorithm that, given a proof-tree PT for $\tau = \alpha_1 \to \ldots \to \alpha_k \to A$, $k \geq 1$, produces (in a non-deterministic way) a complete subtree of FT_τ. If there is a primitive part $p_{\langle i\rangle} = \boxed{\begin{smallmatrix}A\\ |\end{smallmatrix}}$ for some $1 \leq i \leq k$, then



is a complete subtree of FT_τ. Otherwise, for every primitive part $p_{\iota'}$ of form (P3), i.e. $p_{\iota'} = \boxed{\begin{smallmatrix}A\\ |\end{smallmatrix}}$, there is $\iota' = \iota \cdot \langle i, j\rangle$ and $p_{\iota'}$ descends from the ith tail-variable of primitive part p_ι in FT_τ. Now, proceed as follows:

(a) If there is an occurrence of a primitive part $p_{\iota\cdot\langle i,j\rangle}$ that in PT does not descend from the ith tail-variable of an occurrence of p_ι, then we know from the third condition on proof-trees that there is at least one occurrence of p_ι such that the occurrence of $p_{\iota\cdot\langle i,j\rangle}$ is in the subtree rooted in the ith tail-variable of p_ι. Substitute this whole subtree by $p_{\iota\cdot\langle i,j\rangle}$. The result is clearly still a proof-tree for τ. Name it PT and repeat as long as there are primitive parts of form (P3) meeting the condition above.

(b) Now, there is at least one subtree of PT of the form

$$\begin{tikzcd} & p_\iota \arrow[dl, Rightarrow, no head] \arrow[dr, Rightarrow, no head] & \\ p_{\iota_1} & \ldots & p_{\iota_n} \end{tikzcd} ,$$

with $n \geq 1$ and such that for $1 \leq i \leq n$ each p_{ι_i} is of form (P3) and descends from the ith tail-variable in FT_τ. If $|\iota| = 1$, then

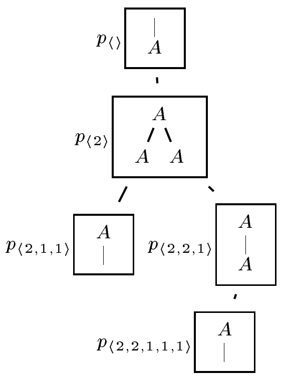
$$\begin{tikzcd} & p_{\langle\rangle} \arrow[d, no head] & \\ & p_\iota \arrow[dl, dashed, no head] \arrow[dr, dashed, no head] & \\ p_{\iota_1} & \ldots & p_{\iota_n} \end{tikzcd}$$

is a complete subtree of FT_τ. Otherwise, repeat step a. as long as possible to subtrees of this form (instead of subtrees consisting of a primitive part of form (P3)). Since the number of primitive parts in PT decreases in every step, this procedure eventually stops producing a complete subtree of FT_τ.

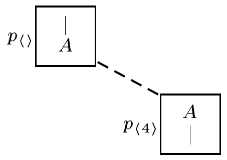
∎

COROLLARY 5.20
A monatomic formula $\tau = \alpha_1 \to \ldots \to \alpha_k \to A$, where $k \geq 1$, is provable iff $\alpha_i \to A$ is provable for some $1 \leq i \leq k$.

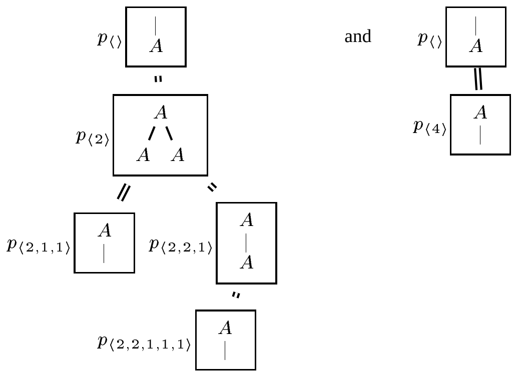
EXAMPLE 5.21
The following is a complete subtree obtained by pruning $\texttt{tree}(\tau)$ of Example 5.17.


$p_{\langle\rangle}$
A
$p_{\langle 2\rangle}$
A
A A
$p_{\langle 2,1,1\rangle}$
A
$p_{\langle 2,2,1\rangle}$
A
A
$p_{\langle 2,2,1,1,1\rangle}$
A

and another one is



The corresponding proof-trees in the proof of Theorem 5.19 are



Consequently, τ is provable and has an infinite number of normal proofs.

Note that from any, possibly non-monatomic, provable formula τ one obtains a provable monatomic formula by instantiation. Thus, we get the following necessary condition for provability of formulas in general from Theorem 5.19.

COROLLARY 5.22
For every provable formula τ at least one complete subtree can be obtained by pruning FT_{τ}.

Acknowledgements

The work presented in this paper has been partially supported by funds granted to *LIACC* through *Programma de Financiamento Plurianual, Fundação para a Ciência e Tecnologia* and *Programa POSI*.

References

[1] T. Aoto, and H. Ono. Uniqueness of normal proofs in $\{\rightarrow, \wedge\}$-fragment of NJ. Research Report IS-RR-94-0024F, School of Information Science, JAIST.

[2] T. Aoto. Uniqueness of normal proofs in implicational intuitionistic Logic. Journal of Logic, Language & Information, **8**, 217–247, 1999.

[3] H. Barendregt. Lambda calculi with types. In *Background: Computational Structures*, volume 2 of *Handbook of Logic in Computer Science*, Abramsky, Gabbay, and Maibaum, eds, pp. 117–309. Oxford Science Publications, 1992.

[4] C.-B. Ben-Yelles. *Type-assignment in the Lambda-calculus; Syntax and Semantics*. PhD thesis, Mathematics Dept., University of Wales Swansea, UK, 1979.

[5] S. Broda and L. Damas. Counting a type's principal inhabitants. Proceedings of the 4th International Conference on Typed Lambda Calculi and Applications, *TLCA'99*, pp. 69–82. *Lecture Notes in Computer Science* 1581, Springer Verlag, 1999.

[6] S. Broda and L. Damas. Counting a type's (principal) inhabitants. *Fundamenta Informaticae*, **45**, 33–51, 2001.

[7] S. Broda, and L. Damas. On the structure of normal λ-terms having a certain type. *Proceedings of 7th WoLLIC'2000*, pp. 33–43, 2000.

[8] S. Broda and L. Damas. A context-free grammar representation for normal inhabitants of types in TA_λ. In *Progress in Artificial Intelligence - 10th Portuguese Conference on Artificial Intelligence*, P. Brazdil and A. Jorge, eds, pp. 321–334. *Lecture Notes in Artificial Intelligence*2258, Springer-Verlag, 2001.

[9] S. Broda, and L. Damas. Studying provability in implicational intuitionistic logic: the formula tree approach. *Electronic Notes in Theoretical Computer Science (Elsevier)*, Volume 67, *Proceedings of the 9th Workshop on Logic, Language, Information and Computation (WoLLIC'2002)*, 2002.

[10] S. Broda, L. Damas, M. Finger, and P. Silva e Silva. Decidability of a fragment of $BB'IW$-logic. *Theoretical Computer Science*, **318**, 373–408, 2004.

[11] M.W. Bunder. Proof finding algorithms for implicational logics. *Theoretical Computer Science*, **232**, 165–186, 2000.

[12] M.W. Bunder. Uniqueness of inhabitants of balanced types. Personal communication.

[13] I. Guessarian. *Algebraic Semantics*. *Lecture Notes in Computer Science*, Vol. 99. Springer Verlag, 1981.

[14] J. R. Hindley. *Basic Simple Type Theory*. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 1997.

[15] S. Hirokawa. BCK-formulas having unique proofs. In *Category Theory and Computer Science*, D.H. Pitt *et al.*, eds, pp. 106–120. *Lecture Notes in Computer Science* 530, Springer-Verlag, 1991.

[16] S. Hirokawa. Principal types of BCK-lambda-terms. *Theoretical Computer Science*, **107**, 253–276, 1993.

[17] S. Hirokawa. Note: Infiniteness of proof(α) is polynomial-space complete. *Theoretical Computer Science*, **206**, 331–339, 1998.

[18] W. A. Howard. The formulae-as-types notion of construction. In *To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, J. P. Seldin and J. R. Hindley, eds, pp. 479–490. Academic Press, 1980.

[19] Y. Komori. BCK algebras and lambda calculus. *Proceedings of 10th Symmposium on Semigroups*, pp. 5–11, 1986.

[20] G. E. Mints. A simple proof of the coherence theorem for cartesian closed categories. In *Selected Papers in Proof Theory*, G. E. Mints ed., pp. 213–220. Bibliopolis, 1992.

[21] D. Prawitz. *Natural Deduction*. Almqvist and Wiksell, Sweden, 1965.

[22] M. Takahashi, Y. Akama, and S. Hirokawa. Normal Proofs and Their Grammar. *Information and Computation*, **125**, 144–153, 1996.

[23] M. Tatsuta. Uniqueness of normal proofs of minimal formulas. *Journal of Symbolic Logic*, **58**, 789–799, 1993.

[24] R. Statman. Intuitionistic propositional logic is polynomial-space complete. *Theoretical Computer Science*, **9**, 67–72, 1979.

[25] M. Wand. A simple algorithm and proof for type inference. *Fundamenta Informaticae*, **10**, 115–122, 1987.

Appendix

A Proof of Proposition 3.12

Consider a standard long normal inhabitant N of a type σ and let $M_1, M_2 \in \mathcal{C}(N)$. It follows directly from the definition of algorithm $\mathcal{C}(\_)$ that

1. M_1 and M_2 have exactly the same structure;
2. variables in the same position in abstraction sequences in M_1 and M_2 have exactly the same name x_ι^i for some sequence ι and integer $i \geq 0$;
3. variables in the same position in M_1 and M_2, but not in abstraction sequences, are respectively of the form $x_\iota^{i_1}$ and $x_\iota^{i_2}$ for some sequence ι and integers $i_1, i_2 \geq 0$.

In order to show that M_1 and M_2 have the same principal type we recall an algorithm due to Mitchell Wand [25] that determines the principal type of λ-terms by a finite set of equations between types. When applying the algorithm to a closed normal form M, the set of equations $E(M) = E(\emptyset, \emptyset, M, A)$, where A is a type-variable, defines the principal type of M, up to renaming of type-variables. Given a context Γ and a set of equations between types Σ, the set $E(\Gamma, \Sigma, M, A)$ is computed by the rules below[7], where $\Gamma(x)$ denotes the type assigned to x in Γ.

$$\begin{array}{rcl}
E(\Gamma,\Sigma,\lambda x^{i_1}_{\iota_1}\dots x^{i_n}_{\iota_n}.x^i_\iota M_1\dots M_k,A) & = & E(\Gamma',\Sigma',x^i_\iota M_1\dots M_k,B)\\
 & & \text{where } \Gamma'=\Gamma\cup\{x^{i_1}_{\iota_1}:A_1,\dots,x^{i_1}_{\iota_n}:A_n\}\\
 & & \Sigma'=\Sigma\cup\{A=A_1\to\dots\to A_n\to B\}\\
 & & A_1,\dots,A_n,B \text{ a new variables}\\
 & & \text{and } k\geq 0;\\
E(\Gamma,\Sigma,x^i_\iota,A) & = & \Sigma\cup\{A=\Gamma(x^i_\iota)\};\\
E(\Gamma,\Sigma,x^i_\iota M_1\dots M_k,A) & = & \bigcup_{1\leq j\leq k}E(\Gamma,\Sigma',M_j,B_j),\\
 & & \text{where}\\
 & & \Sigma'=\Sigma\cup\{\Gamma(x^i_\iota)=B_1\to\dots\to B_k\to A\}\\
 & & \text{and } B_1,\dots,B_k \text{ are new variables.}
\end{array}$$

So, in order to show that the terms M_1 and M_2 have the same principal type it suffices to show that $E(M_1)$ and $E(M_2)$ are equivalent sets of equations. It follows from Conditions 1–3 in the beginning of the proof, that the steps performed during the computation of $E(M_1)$ and $E(M_2)$ are identical, that the bases Γ used in these steps are identical and that the equations generated in these steps are identical but eventually for type-variables $\Gamma(x^{i_1}_\iota)$ (resp. $\Gamma(x^{i_2}_\iota)$) in them, which might be distinct if $i_1\neq i_2$. Now, consider a step in the computation of $E(M_1)$ where a rule $E(\Gamma,\Sigma_1,X_1,A)$ is applied to a subterm X_1 of M_1, which is no abstraction and has head-variable $x^{i_1}_\iota$, for some sequence ι and integer $i_1\geq 0$. Also, consider the corresponding step in the computation of $E(M_2)$, where the same rule $E(\Gamma,\Sigma_2,X_2,A)$ is applied to the corresponding subterm X_2 of M_2, which has by Condition 3 head-variable $x^{i_2}_\iota$, for some integer $i_2\geq 0$. Clearly, one has $x^{i_1}_\iota:A_1\in\Gamma$ and $x^{i_2}_\iota:A_2\in\Gamma$, where $A_1\neq A_2$ iff $i_1\neq i_2$. Now, it suffices to show that $\Sigma_1\models A_1=A_2$[8] as well as $\Sigma_2\models A_1=A_2$.

We proceed by induction on the length of ι. In fact, it follows from Lemma 3.16 and from the definition of $\mathcal{C}(\_)$ that the initial (outer-most) abstraction sequence of M_1 is of the form $\lambda x^0_{\langle 1\rangle}\dots x^0_{\langle n\rangle}$, for some $n\geq 1$, and that no other abstraction sequence in M_1 contains variables indexed by a sequence ι of length 1. So, if $|\iota|=1$, then $i_1=i_2=0$, thus $x^{i_1}_\iota=x^{i_2}_\iota$, and consequently $A_1=A_2$. Now, consider a sequence ι of the form $\iota'\cdot\langle i,j\rangle$. It follows from Lemma 3.10 and from $x^{i_1}_\iota:A_1\in\Gamma$ and $x^{i_2}_\iota:A_2\in\Gamma$, that there are two subterms T_1 and T_2 of M_1, respectively of the form $T_1=x^{l_1}_{\iota'}N^1_1\dots N^1_m$ and $T_2=x^{l_1}_{\iota'}N^2_1\dots N^2_m$, such that $x^{i_1}_{\iota'\cdot\langle i,j\rangle}(=x^{i_1}_\iota)$ is the jth variable in the initial abstraction sequence of N^1_i and $x^{i_2}_{\iota'\cdot\langle i,j\rangle}$ is the jth variable in the initial abstraction sequence of N^2_i. Furthermore, either X_1 is a subterm of T_1 and T_1 is a subterm of N^2_i, or X_1 is a subterm of T_2 and T_2 is a subterm of N^1_i. In both cases, one has $x^{l_1}_{\iota'}:B_1\in\Gamma$ and $x^{l_2}_{\iota'}:B_2\in\Gamma$, for some variables B_1 and B_2. Thus, by the induction hypothesis $\Sigma_1\models B_1=B_2$. Also, it follows from the definition of the algorithm $E(\_,\_,\_,\_)$ that Σ_1 contains equations of the form

$$\Gamma(x^{l_1}_{\iota'})=B^1_1\to\dots\to B^1_m\to A^1 \tag{A.1}$$

$$\Gamma(x^{l_2}_{\iota'})=B^2_1\to\dots\to B^2_m\to A^2 \tag{A.2}$$

where $B^1_1,\dots,B^1_m,A^1,B^2_1,\dots,B^2_m,A^2$ are type-variables. Furthermore, Σ_1 contains equations of the form

$$B^1_i=C^1_1\to\dots\to C^1_{j-1}\to A_1\to C^1_{j+1}\to\dots\to C^1_{k_i}\to D^1_i \tag{A.3}$$

$$B^2_i=C^2_1\to\dots\to C^2_{j-1}\to A_2\to C^2_{j+1}\to\dots\to C^2_{k_i}\to D^2_i \tag{A.4}$$

for some type-variables $C^1_1,\dots,C^1_{j-1},C^1_{j+1},\dots,C^1_{k_i},D^1_i,C^2_1,\dots,C^2_{j-1}$, $C^2_{j+1},\dots,C^2_{k_i},D^2_i$. Note again that $\Sigma_1\models\Gamma(x^{l_1}_{\iota'})=\Gamma(x^{l_2}_{\iota'})$ follows from the induction hypothesis. Thus, we conclude from Equations (A.1) and (A.2) that $\Sigma_1\models A^1=A^2$ and $\Sigma_1\models B^1_s=B^2_s$ for $1\leq s\leq m$. In particular, $\Sigma_1\models B^1_i=B^2_i$ and consequently Equations (A.3) and (A.4) imply that $\Sigma_1\models A_1=A_2$. The proof of $\Sigma_2\models A_1=A_2$ is identical. ∎

Received 2 August 2004

[7]In this appendix we let $A,B,\dots$ denote any type and not only atoms.

[8]We write $\Sigma\models A=B$ to mean that $A=B$ is a consequence of the equalities in Σ.