A new name-free system of lambda calculus with delayed updating

6th draft version

Ferruccio Guidi and Rob Nederpelt

February 27, 2024

Abstract

In this paper we focus on the updating of variables connected to beta-reduction in lambda calculus.

1 Introduction

It is well known that practical implementations of λ-calculus turn α-equivalence into syntactic equality by representing bound variable occurrences with numbers rather than names (de Bruijn, 1972). Generally speaking, function application involves replacing the occurrences of the independent variables in the function's body with copies of the function's arguments and, in this scenario, the indexes occurring in such copies may need an update to prevent captures. Experience shows that this update, known as lift according to a well-established terminology, is a time consuming operation (Guidi, 2009, Appendix A2) that, precisely, computing machines strive to avoid (Kluge, 2005). v5. 1–1

In a family of systems originating from de Bruijn (1978b) and $\lambda\sigma$ (Abadi *et al.*, 1991), β-reductions insert information on updates in the copied terms with depth indexes or level indexes (de Bruijn, 1972), being positive numbers. Therefore, the resulting systems are termed *name-free* as opposed to *name-carrying*. Thus a computation can delay updates at will or apply them whenever is the case. v5. 1–2

The name-free system we present in this article for the first time lies in this family. Contrary to its predecessors, it is based just on β-reduction at a distance, an extension of β-reduction introduced in Nederpelt (1973). Such a reduction relation allows, for example, not only β-reduction $K \equiv (\lambda xy\,.\,L)MN \to (\lambda x\,.\,L[y := M])N$, but also $K \to (\lambda y\,.\,L[x := N])M$.

In deviation of the usual notation, lambda terms are presented as trees composed of branches. With this representation, one obtains a transparent view on matching pairs of abstraction and application, each of which pairs may generate a beta-reduction. This transparency facilitates our discussion considerably.

The name-free system of lambda-calculus with which the paper concludes has the following properties:

– Name-free variables are positive integers, as usual, and the employed reduction is β-reduction at a distance.

– None of the variables disappears by a β-reduction: name-free variables have a fixed position in the lambda term, not altered by the reduction of the term.

– As a consequence, some variables become embedded *inside* the term by β-reduction; such variables retain information on the delayed update.

– If a lambda term β-reduces to another one, then the first term is *included* in the second one, i.e., the tree of the first term is a *subtree* of the tree of its reduct.

– Each sequence of one-step β-reductions in the traditional lambda calculus corresponds one-to-one to a similar sequence in the new system, and vice versa.

– A kind of 'garbage collection' reduces a lambda term in the new system to one in the usual lambda calculus.

– The described β-reduction obeys confluence.

– Weak normalization of a lambda term in the new system implies strong normalization.

We divide our exposition into six parts. In Section 2 we give a concise historic overview of updating in name-free beta reduction. We recall name-carrying λ-calculus in Section 3 and name-free λ-calculus in Section 4. The main part of the paper is Section 5. Here we describe our new system with delayed updating, for the case of name-free variables, and we compare the system with traditional lambda calculus. Our concluding remarks are in Section 6. In Appendix **??** we present a variant of β-reduction for our new system, while in Appendix **??** we shortly describe our system for the name-*carrying* case.

2 A short history of updating in name-free beta-reduction

In name-carrying systems of λ-calculus a binder of a term M, say λ_x, and the variable occurrences that refer to it carry the same name, say x. On the contrary, name-free systems use unnamed binders, say λ, and replace a bound variable occurrence x with an index that is a non-negative integer denoting the position of the corresponding λ_x along the path connecting x to the root of M in the representation of M as an abstract syntax tree. As we pointed out in the introduction, the β-reduction step of the latter systems requires updating the indexes occurring in a copied argument, say N, to maintain the relationship between the bound variable instances in M and the respective binders. Depending on the particular system, if *immediate updating* is in effect, the update occurs by applying a so-called *update function* to the indexes in N. On the contrary, if *delayed updating* is in effect, the update function is just stored in the syntax of the copied N. v5. 2–1 v5. 2–2

The first name-free systems with immediate updating appear in de Bruijn (1972) with the basic update functions $\tau_{d,h}$ of type $\mathbb{N}^+ \to \mathbb{N}^+$, where $d \in \mathbb{N}$ and $h \in \mathbb{N}$.

$$\tau_{d,h} \equiv i \mapsto \begin{cases} i & \text{if } i \leq d \\ i+h & \text{if } i > d \end{cases}$$

The systems accompanying de Bruijn (1978b) – for instance those in de Bruijn (1977, 1978a) – are the first to allow delayed updating by featuring the term node $\phi(f)$ where f is an arbitrary function of type $\mathbb{N}^+ \to \mathbb{N}^+$. The original purpose of $\phi(f)$ is to present substitution as a single operation defined by recursion on the structure of terms. v5. 2–3 v5. 2–4

Other systems of the same family, like Nederpelt (1979, 1980); Kamareddine and Nederpelt (1993), feature the term node $\mu(d,h)$ or $\phi^{(d,h)}$ that holds the function $\tau_{d,h}$. Moreover, the systems originating from Abadi *et al.* (1991) – for instance those in Curien, Hardin and Lévy, 1996 – feature the explicit substitution constructors *id* and $\uparrow$ that essentially hold the functions $\tau_{0,0}$ (the identity) and $\tau_{0,1}$ (the successor) respectively. v5. 2–5

The system we are going to introduce takes a simpler approach. Its syntax has a term node p we call *inner numeric label* where $p \in \mathbb{N}^+$. An active p holds the function $\tau_{0,p}$ while a passive, i.e. present but ignored, p holds the function $\tau_{0,0}$. In some sense, we want to show that supporting the functions $\tau_{0,h}$ suffices to implement delayed updating in basic name-free λ-calculus.

3 Name-carrying lambda trees

3.1 Lambda-trees

We use a tree representation for lambda terms in (untyped or typed) lambda calculus. See Figure 1. The obtained tree is an undirected rooted tree with labels attached to the *edges* of the tree. We prefer labelling the edges instead of the vertices for reasons to be explained below.

Label L represents a 'lambda' and A stands for an 'application'. Moreover, we use label S for 'subordination'; this extra label enables us to discriminate between main branches and subbranches, which is essential when considering different pathes in a tree.

We assume that an infinite set of *variables*, $\mathcal{V}$, is available.

Definition 3.1. *tree*: connected acyclic undirected graph, consisting of vertices and labelled edges.

labels: symbols L_x (for each variable $x \in V$), A, S, or a variable name. v5. 3–1

path: a connected string of labelled edges occurring in a tree.

Definition 3.2. *Meta-variables for trees*: $\mathbf{t}$, $\mathbf{t}'$, $\mathbf{t}_1$, ...; *for labels*: ℓ, ...; *for paths*: p, q, ...

Convention 3.3. *Each* path *in a tree will be* identified *with its string of labels.*

Definition 3.4. Let p and q be paths. Then $p \preceq q$ if there is a (possibly empty) path p' obeying $q \equiv p\,p'$, and $p \prec q$ if there is a non-empty path p' with that property.

Definition 3.5. A-cell, L-cell, var-cell: see Figure 1.

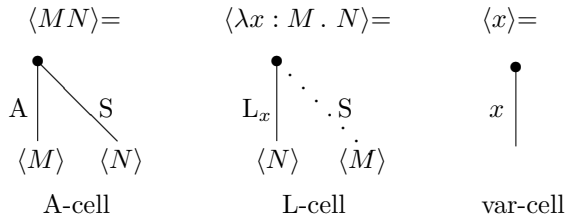

Figure 1: cells of lambda trees

Explanation 3.6. (*i*) *An* A-cell *initializes an* application*, of* function M *to* argument N*. Below the* A*-edge the tree representation of function* M *must be attached, below the* S*-edge that of the argument* N*. Clearly, an* A*-cell is* binary*.* v5. 3–2

(*ii*) *Each* L-cell *starts an* abstraction*, abstracting the* function body N*, possibly containing* x *as a free variable, over that* x*. Below the* L_x*-edge, the representation of the body* N *is expected. This kind of cell depends on the name of the variable. That is to say, for another variable, say* y*, an* L_y*-cell can be constructed in a similar manner.*

As to the S*-edge in an* L*-cell, and the* type M *of* x*, there are two options.*

– *In* untyped λ*-calculus, a variable* x *has no type at all, so the type* M *is absent. Consequently, the* S*-edge should be suppressed completely (indicated by the dots in the drawn* S*-edge).*

– *In the* typed λ*-calculus, however, there must be an* S*-edge, enabling one to attach the type* M *of* x*.*

Hence, an L*-cell is either unary (in the untyped case) or binary (in the typed case).*

(*iii*) *A* var-cell *represents a single variable. Such a cell is always unary. A var-cell depends on the variable being the label. So there is an infinity of different var-cells.*

Now we comment on how these three kinds of cells are *used* to construct so-called λ*-trees*, which are aimed to be structured representations of λ-terms.

We first note that the roots of cells are always vertices. But at the lower sides, A-cells and L-cells have so-called *open ends*. The idea is, that these open ends enable to connect these cells *downwards* to roots of other A- or L-cells or the root of a var-cell; and *upwards* to open ends of other A- and L-cells.

Note, however, that var-cells can only be *upwards* connected to A- or L-cells, but they can not be *downwards* connected to any other cell: a var-cell does *not* have an open end.

Definition 3.7. A λ-*tree* is a non-empty, rooted and edge-labelled tree, built by an arbitrary connection of L-cells, A-cells and var-cells, such that there are no open ends. (So every leaf is a var-cell and var-cells appear only at the leaves.)

Each λ-tree apparently has a top-cell. The *root* of a λ-tree is the vertex of its top-cell.

Note 3.8. *In many* typed *lambda calculi, one has – separated from the variable set $\mathcal{V}$ – a set of* type variables*, say $\mathcal{T}$. Often there also is a binder for these type variables, usually denoted by symbol* Π*. Moreover, in these calculi one may have one or more* constants*, such as $*$, which denotes the type of type variables.* v5. 3–3

If we want to incorporate such kinds of devices into our tree representation of λ-terms, we have to extend the set of labels with e.g. P_x *(for each type variable x) to represent* Π*, and add names of constants to our labels, e.g. $*$.*

Next, we have to add P*-cells and, e.g., $*$-cells, to our syntactic tools. It will be obvious how to implement these new cells into our system.*

The following example shows how a term from *typed* lambda calculus can be pictured by means of the mentioned cells. Note that we also need a P-cell and $*$-cells (cf. Note 3.8).

Example 3.9. In Figure 2 (1) we picture the λ-tree of the following term from typed lambda calculus:

$\lambda\alpha : * \,.\, \lambda\beta : * \,.\, \lambda x : \alpha \,.\, \lambda y : \alpha \to \beta \,.\, y\, x.$

The tree marked (1) is written in the *name-carrying* notation, using the *variable names* x, y, α and β. The notation $\alpha \to \beta$ is treated as an abbreviation of $\Pi u : \alpha \,.\, \beta$.

(The tree in Figure 2 (2) will be discussed later.) v5. 3–4

Paths in λ-trees are essential elements of our forthcoming discussion, in particular regarding several kinds of *reduction*. We give names to particular paths in the following definition.

Definition 3.10. Let $\mathbf{t}$ be a λ-tree.

We write $p \in \mathbf{t}$ if path p is a (coherent) part of $\mathbf{t}$.

A *root path* p in $\mathbf{t}$ is a (non-empty) path starting in the root. Notation: $p \in^{\wedge} \mathbf{t}$.

A *leaf path* in $\mathbf{t}$ is a (non-empty) path with a leaf as final label. Notation: $p \in_{\vee} \mathbf{t}$.

A *complete path* in $\mathbf{t}$ is a path that is both a root path and a leaf path. Notation: $p \in^{\wedge}_{\vee} \mathbf{t}$.

Example 3.11. In the λ-tree of Example 2 (1), the path $\mathrm{L}_\alpha \,\mathrm{L}_\beta \,\mathrm{L}_x \,\mathrm{S}\,\mathrm{P}_u \,\beta$ is a complete path.

For the sake of convenience, we assume that the binding variables in a λ-tree always differ from each other. Let's assume, for example, that we consider typed lambda calculus and that the set $\mathcal{B}$ of all binders is $\{\mathrm{L}_x | x \in \mathcal{V}\} \cup \{\mathrm{P}_x | x \in \mathcal{T}\}$. Then we obey the following convention.

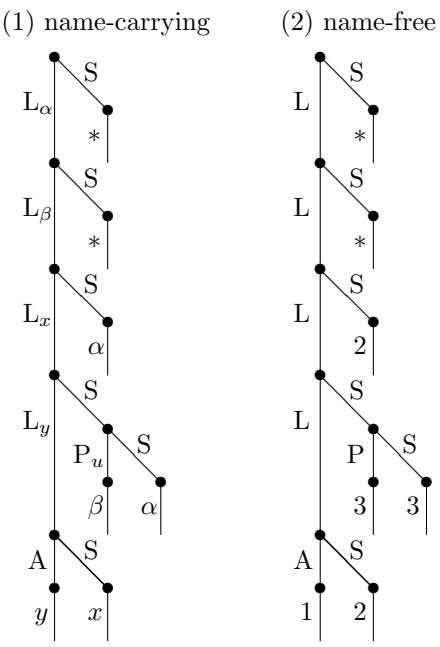

Figure 2: Lambda trees

Convention 3.12. *Let* $\mathbf{t}$ *be a* λ*-tree, let* B_x *and* $B'_y \in \mathcal{B}$ *and let the paths* $p\,B_x$ *and* $q\,B'_y \in^\wedge \mathbf{t}$*. Then* $x \equiv y \;\Rightarrow\; (p \equiv q \wedge B \equiv B')$.

Hence, in particular: If B_x is a binder in $\mathbf{t}$, then there is no other binder with subscript x in $\mathbf{t}$.

Definition 3.13. (*i*) Let $p_1, \ldots, p_n$ be all complete paths in a λ-tree $\mathbf{t}$. Then we identify $\mathbf{t}$ with the set $\{p_1, \ldots, p_n\}$.

(*ii*) Let $\mathbf{t} \equiv \{p_1, \ldots, p_m\}$ and $\mathbf{t}' \equiv \{q_1, \ldots, q_n\}$ be λ-trees. Then $\mathbf{t}$ is a *subtree* of $\mathbf{t}'$, or $\mathbf{t} \subseteq \mathbf{t}'$, if for each $p_i \in^\wedge_\vee \mathbf{t}$ there exists a path $q_j \in^\wedge_\vee \mathbf{t}'$ such that $p_i \preceq q_j$.

Remark 3.14. *From now on, we focus on the* **untyped** *version of* λ*-calculus, since we concentrate on various forms of* β*-reduction and types play no role in* β*-reduction. Most of the material presented here below, however, also applies to the* typed *version of* λ*-calculus, with some adaptations.*

3.2 β-reduction on λ-trees

The relation β-reduction between λ-terms in typed lambda calculi is defined as the compatible relation generated by

$(\lambda x : K\,.\,M)P \to_\beta M[x := P]$.

Here $M[x := P]$ is the result of substituting P for all free x's in M. (We do not give the necessary conditions on the names of variables which prevent undesired bindings in the 'process' of β-reduction.)

We give another description of β-reduction, suited to the *tree format* sketched previously and facilitating the alternative β-reductions described below. We first give some definitions, and a lemma on binding.

Definition 3.15. Let $\mathbf{t}$ be a λ-tree and $p \in^\wedge \mathbf{t}$.

Consider the set $S \subseteq \mathbf{t}$ of all complete paths $p\,q\,x \in^\wedge_\vee \mathbf{t}$, so the paths beginning with p, ending in some leaf x and having some path q in between. The set of all these $q\,x$ is itself a tree, called *tree*(p).

We call the set S the *grafted tree* of p in $\mathbf{t}$. We denote this grafted tree by $p\,\mathbf{t}'$, if $\mathbf{t}'$ is *tree*(p).

Definition 3.16. Let $\mathbf{t}$ and $\mathbf{t}'$ be λ-trees.

(1) Let $p\,y \in^\wedge_\vee \mathbf{t}$. Then $(p\,y)[x := \mathbf{t}']$ is defined as the grafted tree $p\,\mathbf{t}'$ if $x \equiv y$, and as $p\,y$ if $x \not\equiv y$.

(2) We define $\mathbf{t}[x := \mathbf{t}']$ as $\{q[x := \mathbf{t}'] \mid q \in^\wedge_\vee \mathbf{t}\}$.

(3) Let $p \in^\wedge \mathbf{t}$ but $p \notin^\wedge_\vee \mathbf{t}$. Then $\mathbf{t}[\mathit{tree}(p) := \mathbf{t}']$ is the tree obtained from $\mathbf{t}$ by replacing the grafted tree $p\,\mathit{tree}(p)$ by the grafted tree $p\,\mathbf{t}'$.

Lemma 3.17. *Let* $\mathbf{t}$ *be a* λ*-tree.*

(i) *Assume that* $p\,x \in^\wedge_\vee \mathbf{t}$*. If* x *is a variable in* $\mathbf{t}$ *that is bound in the original* λ*-term by a* λ*, then there is exactly one* $\mathrm{L}_x \in \mathbf{t}$ *binding this* x*, and* $p \equiv p_1\,\mathrm{L}_x\,p_2$ *for some paths* p_1 *and* p_2*.*

(ii) *Assume that* $p\,\mathrm{L}_x \in^\wedge \mathbf{t}$*. If* $p\,\mathrm{L}_x\,q\,x \in^\wedge_\vee \mathbf{t}$*, then* x *is bound by* L_x*. In a* bound *term, all* x*'s are bound by this* L_x*, so there are no* x*'s 'outside'* *tree*$(p\,\mathrm{L}_x)$*.*

Hence, the binder of a variable x in $\mathbf{t}$ can be found on the path leading backwards from x to the root of $\mathbf{t}$. (A similar lemma holds for variables bound by a Π in the original λ-term.) Moreover, an L_x in $\mathbf{t}$ binds all x's that occur in the tree 'below' it. v5. 3–5

Definition 3.18. (i) Let $\mathbf{t}$ be a λ-tree and let $p\,\mathrm{L}_x\,q\,x \in^\wedge_\vee \mathbf{t}$, such that L_x binds x. Then the path $\mathrm{L}_x\,q\,x$ is called the **L-block** *of (this occurrence of)* x.

(ii) A λ-tree is called *closed* if all leaf variables x are bound by an L_x.

Obviously, in a closed λ-tree, every leaf variable x corresponds to exactly one L-block ending in that x.

We can describe *β-reduction* of a λ-tree $\mathbf{t}$, with relation symbol $\to_\beta$, as follows.

Definition 3.19. Let $\mathbf{t}$ be a λ-tree and assume that $p\,\mathrm{A}\,\mathrm{L}_x \in^\wedge \mathbf{t}$.

Now $\mathbf{t} \;\to_\beta\; \mathbf{t}[\mathit{tree}(p) := \mathit{tree}(p\,\mathrm{A}\,\mathrm{L}_x)[x := \mathit{tree}(p\,\mathrm{S})]]$.

Again, we disregard some necessary conditions on the variable names. Note: if $p\,\mathrm{A}\,\mathrm{L}_x \in^\wedge \mathbf{t}$, then also $p\,\mathrm{S} \in^\wedge \mathbf{t}$.

3.3 Some variants of beta-reduction

3.3.1 Balanced beta-reduction

There is a variant of β-reduction that is interesting for certain purposes. We call it *balanced β-reduction*. In the literature, it originally appeared under the name β_1 (Nederpelt, 1973). For details, see the more recent literature about the Linear Substitution Calculus (cf., Accattoli and Kesner, 2010) and Accattoli and Kesner, 2012, in which it is called *distant beta*, symbol $\rightarrow_{dB}$. See also Barenbaum and Bonelli, 2017.

Firstly, we give the following definition.

Definition 3.20. A path p in a λ-tree $\mathbf{t}$ is called *balanced*, denoted $bal(p)$, if it is constructed by means of the following inductive rules:
(i) $bal(\varepsilon)$, i.e., the empty string is balanced;
(ii) if $bal(p)$, then $bal(\mathrm{A}\, p\, \mathrm{L}_x)$, for every variable x;
(iii) if $bal(p)$ and $bal(q)$, then $bal(p\, q)$.
In case (ii), we say that the mentioned A *matches* the mentioned L.

Examples of balanced paths: ε, A L, A A L L, A L A L, A A L A A L L L.

Note the close correspondence between nested paths and (consecutive) nested pairs of parentheses. Only A- and L-labels occur in balanced paths, so there is no other label involved, such as S.

We define *balanced β-reduction*, with symbol $\rightarrow_b$, as follows.

Definition 3.21. Let $\mathbf{t}$ be a λ-tree, let $b \in \mathbf{t}$ be a balanced path and assume that $p\, \mathrm{A}\, b\, \mathrm{L}_x \in^{\wedge} \mathbf{t}$ and there is at least one path $p\, \mathrm{A}\, b\, \mathrm{L}_x\, q\, x \in^{\wedge}_{\vee} \mathbf{t}$.
Then $\mathbf{t} \rightarrow_b \mathbf{t}[tree(p\, \mathrm{A}\, b\, \mathrm{L}_x) := tree(p\, \mathrm{A}\, b\, \mathrm{L}_x)[x := tree(p\, S)]]$.

Compare this with Definition 3.19.

Consider two λ-trees $\mathbf{t}$ and $\mathbf{t}'$ such that $\mathbf{t} \rightarrow_b \mathbf{t}'$ as described in Definition 3.21, so each x has been replaced by $tree(p\, S)$ in $tree(p\, \mathrm{A}\, b\, \mathrm{L}_x)$. Now we have that $\mathbf{t}$ is a subtree of $\mathbf{t}'$, provided that we omit all var-labels x in $\mathbf{t}$. So, balanced β-reduction has the property that it *extends* the original underlying tree $\mathbf{t}$ if all instances of variable x are skipped in $\mathbf{t}$. (In the 'tree-like' image of $\mathbf{t}$, we have to skip the edges of the x-labels, as well.)

We now give the usual definition of *redex* (i.e., *red*ucable *ex*pression) and some notions connected with that. Compare this definition with the condition stated in Definition 3.21.

Definition 3.22. Let $\mathbf{t}$ be a λ-tree, let $b \in \mathbf{t}$ be a balanced path and assume that $p\, \mathrm{A}\, b\, \mathrm{L}_x \in^{\wedge} \mathbf{t}$ and there is at least one path $p\, \mathrm{A}\, b\, \mathrm{L}_x\, q\, x \in^{\wedge}_{\vee} \mathbf{t}$.
Then $tree(p)$ is a *redex*, $tree(p\, \mathrm{A}\, b\, \mathrm{L}_x)$ is the *body* of the redex and $tree(p\, S)$ is the *argument* of the redex.

Apart from the L-block as described in Definition 3.18, there are two other kinds of paths that we will call *blocks*: A-*blocks* and r-*blocks*. See the following definition.

Definition 3.23. Let $\mathbf{t}$ be a λ-tree, $b \in \mathbf{t}$ a balanced path and assume that $\mathrm{A}\, b\, \mathrm{L}_x \in \mathbf{t}$. Let $\mathrm{A}\, b\, \mathrm{L}_x\, q\, x$ be a path in $\mathbf{t}$.

(i) The path $\mathrm{A}\, b\, \mathrm{L}_x\, q\, x \in_\vee \mathbf{t}$ (where L_x binds x) is called the **A-block** *of (this occurrence of)* x.

(ii) An A-block $\mathrm{A}\, b\, \mathrm{L}_x\, q\, x \in_\vee \mathbf{t}$ is a front extension of the L-block $\mathrm{L}_x\, q\, x$; the mentioned A-block and the L-block are called *corresponding*.

(iii) The path $\mathrm{A}\, b\, \mathrm{L}_x$ is called the *redex block* or r-block of x.

We now focus on the variable x bound by the root L_x of the body of a redex.

Lemma 3.24. (i) *The* A*-block of a certain* $x \in \mathbf{t}$*, if it exists, is unique, just as the* L*-block of* x *and the* r*-block of* x*.*

(ii) *In a closed term, each var-label* x *corresponds to exactly one* L*-block; but even when the term is closed, not every* L*-block has a corresponding* A*-block.*

(iii) *An* A*-block of a certain* $x \in \mathbf{t}$ *is a join of exactly one* r*-block and exactly one* L*-block, overlapping at the* L_x *binding* x*.*

3.3.2 Focused beta-reduction

Sometimes there is a need for another form of β-reduction. One occasion is when β-reduction is invoked to model *definition unfolding*: then a defined notion occurring in M, say x, is replaced by the definiens, say P. Such an action generally occurs for only one instance of the definiendum x. So instead of replacing *all* occurrences of x in M, one aims at *precisely one* occurrence.

When adapting β-reduction to this situation, there are several things to be considered:

(i) The substitution $[x := P]$ should act on exactly *one* free x in M.

(ii) Hence, x must occur free in M.

(iii) The redex $(\lambda x : K\,.\, M)P$ should remain active after the intended reduction, since there may be other free x's in M which still need a type annotation (i.e., K); moreover, there must remain a possibility to substitute P for one or more of these x's in a later stage of the process.

All this is covered in the following definition of *focused* β-reduction, for which we use the symbol $\to_f$.

Definition 3.25. Let $\mathbf{t}$ be a λ-tree, let $b \in \mathbf{t}$ be a balanced path and assume that $r\, x \in^\wedge_\vee \mathbf{t}$ is a complete path in $\mathbf{t}$ with $r \equiv p\, \mathrm{A}\, b\, \mathrm{L}_x\, q$. We call the balanced reduction identified by p and focussed on the path $q\, x$, a (p, q)*-reduction*. It is defined by

$\mathbf{t} \;\to_f\; \mathbf{t}[r\, x := r\ \mathit{tree}(p\, \mathrm{S})]$.

The possibility of having *balanced* β-reduction is necessary to be able to deal with redexes which otherwise would be forbidden by the maintenance of the redex $(\lambda x : K\,.\, M)P$ after β-reduction, as described in requirement (iii). See the following example.

Example 3.26. We have, in λ-calculus with normal untyped β-reduction:
$(\lambda x\,.\,((\lambda y\,.\,M)Q))P \to_\beta (\lambda x\,.\,(M[y := Q]))P \to_\beta M[y := Q][x := P]$.
In *focused* β-reduction, this becomes:
$(\lambda x\,.\,((\lambda y\,.\,M)Q))P \to_f (\lambda x\,.\,(\lambda y\,.\,M[y_0 := Q])Q)P \to_f$
$(\lambda x\,.\,((\lambda y\,.\,M[y_0 := Q])Q)[x_0 := P])P$.
Here y_0 and x_0 are selected instances of the free y's and x's in M, respectively.
The second of the two one-step, focused, reductions could not be executed without the possibility to have a balanced λ-term $(\lambda y\,.\,M[y_0 := Q])Q$ between the λx and the P.

3.3.3 Erasing reduction

After having applied balanced or focused β-reduction, one also desires a reduction that gets rid of the 'remains', i.e., the A and the L_x in grafted trees $p\,\mathrm{A}\,b\,\mathrm{L}_x\,\mathbf{t}'$ where $x \notin FV(\mathbf{t}')$. Such pairs $\mathrm{A}\ldots\mathrm{L}_x$ are superfluous, since $\mathbf{L}_x$ has no x that is bound to it. We call the corresponding reduction *erasing* β-reduction and use symbol $\to_e$ for it. (This reduction is also referred to as 'garbage collection' in the literature.)

Definition 3.27. Let $\mathbf{t}$ be a λ-tree, let $b \in \mathbf{t}$ be a balanced path and assume that $p\,\mathrm{A}\,b\,\mathrm{L}_x\,\mathbf{t}' \in^\wedge_\vee \mathbf{t}$. Assume moreover that $x \notin FV(\mathbf{t}')$.
Then $\mathbf{t} \ \to_e \ \mathbf{t}[\mathit{tree}(p) := \mathit{tree}(p\,\mathrm{A})[\mathit{tree}(p\,\mathrm{A}\,b) := \mathit{tree}(p\,\mathrm{A}\,b\,\mathrm{L}_x)]]$.

We denote the transitive closure of a reduction $\to_i$ by $\twoheadrightarrow_i$. An arbitrary sequence of reductions $\to_i$ and $\to_j$ is denoted $\twoheadrightarrow_{i,j}$.

Theorem 3.28. *Let* $\mathbf{t}$ *and* $\mathbf{t}'$ *be* λ*-trees.*
(*i*) $\mathbf{t} \twoheadrightarrow_\beta \mathbf{t}' \Rightarrow \mathbf{t} \twoheadrightarrow_{b,e} \mathbf{t}'$.
(*ii*) $\mathbf{t} \twoheadrightarrow_b \mathbf{t}' \Rightarrow \mathbf{t} \twoheadrightarrow_f \mathbf{t}'$
(*iii*) (Postponement of $\to_e$ after $\to_b$) *If* $\mathbf{t} \twoheadrightarrow_{b,e} \mathbf{t}'$, *then there is* $\mathbf{t}''$ *such that* $\mathbf{t} \twoheadrightarrow_b \mathbf{t}'' \twoheadrightarrow_e \mathbf{t}'$.
(*iv*) (Postponement of $\to_e$ after $\to_f$) *If* $\mathbf{t} \twoheadrightarrow_{f,e} \mathbf{t}'$, *then there is* $\mathbf{t}''$ *such that* $\mathbf{t} \twoheadrightarrow_f \mathbf{t}'' \twoheadrightarrow_e \mathbf{t}'$.

Proof (*iii*) Nederpelt, 1973, p. 48, Theorem 6.19.
(*iv*) Similarly. □

Theorem 3.29. $\to_b$, $\to_f$ *and* $\to_e$ *are confluent.*

Proof For $\to_b$ and $\to_e$, see Nederpelt, 1973, Theorems 6.38 and 6.42. For $\to_f$, see Accattoli and Kesner, 2012.

4 Name-free lambda-terms

4.1 The binding of variables

A recurrent nuisance in the formalization of lambda calculus is the *naming* of variables, which plays a dominant role in the establishment of binding. Let's

consider some λ-term in which λx binds variable x. Then one may replace both mentioned x's by a y, on the condition that this is done consistently through the term, and that one prevents that the variable renaming does lead to a name-clash. (This relation is called α-reduction.) For example, the mentioned renaming is forbidden in the term $\lambda x\,.\,\lambda y\,.\,x$, for obvious reasons.

Another cause of worry is that *beta*-reduction has a spreading effect on all kinds of variables, so that it is sometimes a precarious matter to ensure that no 'undesired' binding between a λ and a variable arises.

To prevent these matters, N.G. de Bruijn invented a *name-free* version for terms in λ-calculus (de Bruijn, 1972). Instead of using names such as x to record bindings in terms, he employed *natural numbers*. The idea is to see the λ-term as a tree, comparable to the way we introduced λ-trees in the previous chapter. The principle is, that a variable with number n is bound to the λ which can be found by following the root path ending in this n, and choosing the n-th λ along this path as the binder. In a λ-tree, we call such an end label n a *numerical variable* or a *num-label*.

This approach is known as: 'bound variables references by depth' (de Bruijn, 1978a). Another way of identifying the binder, also discussed in that paper, is to count the number of λ's from the root to the intended one ('reference by level').

In Example 2 (2), the name-carrying λ-tree of Example 2 (1) has been exhibited, as an example of a name-free tree.

This *name-free* representation of λ-calculus, straightforward as it seems, is not so simple as it appears. A pleasant feature of it is that α-reduction is no longer required. But on the other hand, when applying β-reduction, a lot of updating is necessary. This updating is not very easy and it may require extra calculations that complicate matters.

Example 4.1. Consider the term $t_1 \equiv (\lambda x\,.\,\lambda y\,.\,(\lambda z\,.\,y)x)$ in untyped lambda calculus. This term β-reduces to $t_2 \equiv \lambda x\,.\,\lambda y\,.\,y$.

In name-free notation, $t_1 \equiv \lambda\,\lambda\,(\lambda\,2)\,2$. (Note that the third λ is not on the root path of the free x; so, the x in t_1 becomes not 3, but 2 in the name-free version.)

The name-free version of t_2 is $\lambda\,\lambda\,1$: the first number 2 in t_1 must be updated after the β-reduction: it returns as the number 1 in t_2. (The second 2 in t_1, together with the third λ, vanish by the β-reduction.)

Note 4.2. *There is another version of name-free trees for λ-terms, in which the labels are situated not at the edges – as in our proposal – but at the nodes (see de Bruijn, 1978a). Moreover, the labels* S *that we use, are omitted. This works just as well, but for two details. We show this with the same Example 2 (2).*

Imagine the same tree, but with all labels 'raised' to the closest node. So, for example, the root vertex is now labelled L *and the* S*'s have vanished.*

(1) When we consider the tree as not being embedded in the plane, then it is unclear in this case which is the main term and which is the subterm for a given branching. But this can be easily solved by defining trees as planar.

(2) A more serious matter is that an extra provision is necessary for determining the binder of a variable. For example, in the tree of Example 2 (2) with all labels raised, the root path of the variable numbered 4 becomes one L *longer, viz.* L L L L P*. Now observe that the fourth* L *is never meant to bind a variable on this path. So one has to neglect that fourth* L *when counting backwards from 4 to 0 in the process of finding its binder. (Simply changing variable number 4 into 5 is not the proper way to solve this problem; for example, this makes Lemma 3.17, (2), untrue, thus undermining the intended binding structure.)*

Definition 4.3. We use the symbol $\mathcal{T}^{car}$ for the set of *name-carrying*, closed λ-trees. The symbol $\mathcal{T}^{fre}$ means the set of *name-free*, closed λ-trees.

The following sections discuss β-reduction in the name-free case. Since reduction itself is independent of typing, we do not distinguish between typed or untyped version of λ-calculus. So when discussing the sets $\mathcal{T}^{car}$ and $\mathcal{T}^{fre}$ we do not bother whether the terms are typed or not.

4.2 Name-free reductions with instant updating

In the present section we consider name-free β-reduction and some of its variants and each time we describe the updating that is required. We consider β-reduction and its variants, all with *instant updating*. That is, each one-step reduction ends in a λ-tree in which the num-labels have immediately been updated. In Section 5, we try to simplify the name-free β-reduction of λ-terms, by postponing the updates (*delayed updating*).

Note 4.4. *Below, we use the same symbols for the various name-free reductions, as in the* named *case of Section 3.3 (viz., $\to_\beta$, $\to_b$ and $\to_f$). There will be no confusion, since we use letters like* $\mathbf{t}$ *for paths in the name-carrying cases, and* $\mathbf{u}$ *in the name-free cases.*

4.2.1 Beta-reduction

We give a description of β-reduction with instant updating in Definition 4.6.

Notation 4.5. (*i*) *We use the notation $[x := A; y := B]$ for simultaneous substitution.*

(*ii*) *The* lenght $|p|$ *of a path p is the number of labels (including num-labels) in p.*

(*iii*) *The* L-length $\|p\|$ *of a path p is the number of binders (i.e.,* P *or* L*) occurring in p.*

Definition 4.6. Let $\mathbf{u} \in \mathcal{T}^{fre}$ and assume that $p\,\mathrm{A}\,\mathrm{L} \in^\wedge \mathbf{u}$. Then the β-reduction based on the redex identified by p, also called *p-reduction*, is

$\mathbf{u} \to_\beta \mathbf{u}\,[tree(p) := \{q\,n \in^\wedge_\vee tree(p\,\mathrm{A}\,\mathrm{L})\,[upd_1\ ;\ upd_2]\}]$, where

$$\begin{cases} upd_1 \equiv n := n-1 \text{ if } n > \|q\|{+}1, \\ upd_2 \equiv n := \{r\,l \in^\wedge_\vee tree(p\,\mathrm{S})\,[upd_3]\} \text{ if } n = \|q\|{+}1 \end{cases}$$
$$\text{and } upd_3 \equiv l := l + \|q\| \text{ if } l > \|r\|$$

We now explain the contents of this definition.

Just as in Definition 3.19, we consider two subtrees of the original λ-tree $\mathbf{u}$: $tree(p\,\mathrm{A}\,\mathrm{L})$ and $tree(p\,\mathrm{S})$. Both trees need updating because of the performed β-reduction. In the first tree, we have two simultaneous updatings, upd_1 and upd_2. They act on *all* complete paths $q\,n$ in $tree(p\,\mathrm{A}\,\mathrm{L})$, and the choice depends on the value of the leaf n in the path considered. We discern two cases: $n > \|q\|{+}1$ and $n = \|q\|{+}1$. (It is understood that in the case not mentioned, viz. $n < \|q\|{+}1$, the *identical* update is meant, so $n := n$.) See Figure 3 for a visual explanation.

Note that, in the case $n = \|q\|{+}1$, the n is replaced by a copy of the full $tree(p\,S)$, updated by upd_3 if $l > \|r\|$. (Also here, the intention is that an identical update $l := l$ is applied on $r\,l \in^{\wedge}_{\vee} tree(p\,S)$, in the missing case $l \leq \|r\|$.) If $n > \|q\| + 1$, one has to compensate (n becoming $n - 1$) for the missing L.

In all three cases, not only the relevant A-L-pair, but also the grafted tree $\mathrm{S}\,tree(p\,\mathrm{S})$ has vanished.

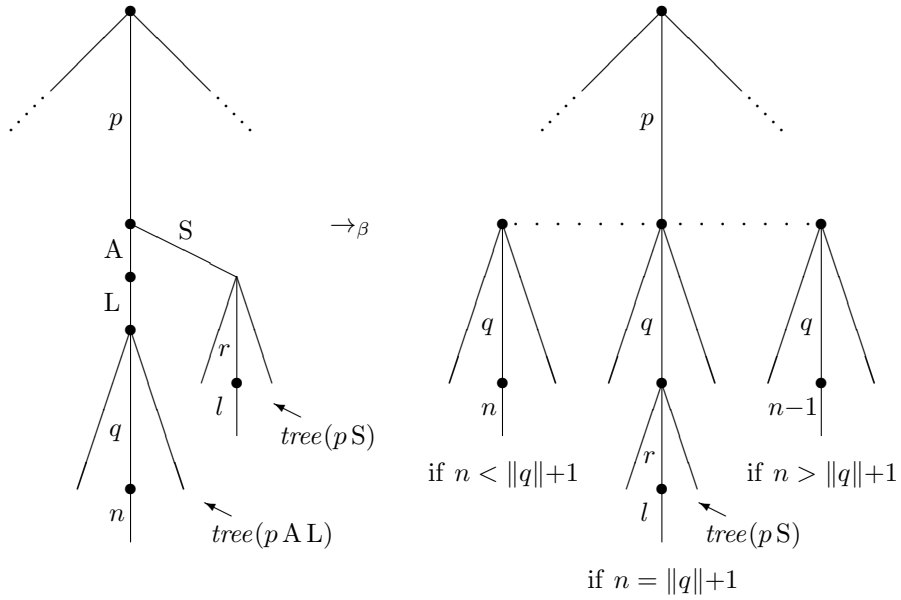

Figure 3: A picture of name-free β-reduction with updating

4.2.2 Balanced beta-reduction

For balanced β-reduction, the updates are somewhat different, due to the non-vanishing of the A-L-couple involved, and the remaining of the original 'argument' $tree(p\,S)$.

Definition 4.7. Let $\mathbf{u} \in \mathcal{T}^{fre}$, let $b \in \mathbf{u}$ be a balanced path and assume that $p\,\mathrm{A}\,b\,\mathrm{L} \in^{\wedge} \mathbf{u}$. Then the balanced reduction based on the redex identified by p (again called p-reduction) is

$\mathbf{u} \to_b \mathbf{u}\,[tree(p\,\mathrm{A}\,b\,\mathrm{L}) := \{q\,n \in^\wedge_\vee tree(p\,\mathrm{A}\,b\,\mathrm{L})\,[upd_1]\}]$, where

$$upd_1 \equiv n := \{r\,l \in^\wedge_\vee tree(p\,\mathrm{S})[upd_2]\}\ \textit{if}\ n = \|q\|{+}1$$
$$\text{and } upd_2 \equiv l := l + \|q\|{+}1 + \|b\|\ \textit{if}\ l > \|r\|$$

As in Section 4.2.1, an identical substitution (i.e., nothing changes) applies in the missing cases for n and l.

4.2.3 Focused beta-reduction

In the case of focused β-reduction, a simple adaptation of Section 4.2.2 is required. This leads to the following.

Definition 4.8. Let $\mathbf{u} \in \mathcal{T}^{fre}$, let $b \in \mathbf{u}$ be a balanced path, assume that $p\,\mathrm{A}\,b\,\mathrm{L} \in^\wedge \mathbf{u}$ and that $q\,n$ is a fixed, complete path in $tree(p\,\mathrm{A}\,b\,\mathrm{L})$.

Consider a β-reduction identified by p. When focusing on q as the *focus path*, we call this a (p,q)-*reduction* defined by

$\mathbf{u} \to_f \mathbf{u}\,[tree(p\,\mathrm{A}\,b\,\mathrm{L}) := tree(p\,\mathrm{A}\,b\,\mathrm{L})[upd_1]]$, where

$$upd_1 \equiv q\,n := q\,\{r\,l \in^\wedge_\vee tree(p\,\mathrm{S})[upd_2]\}\ \textit{if}\ n = \|q\|{+}1$$
$$\text{and } upd_2 \equiv l := l + \|q\|{+}1 + \|b\|\ \textit{if}\ l > \|r\|$$

4.2.4 Erasing reduction

For erasing reduction, the following definition applies.

Definition 4.9. Let $\mathbf{u} \in \mathcal{T}^{fre}$, let $b \in \mathbf{u}$ be a balanced path, assume that $p\,\mathrm{A}\,b\,\mathrm{L} \in^\wedge \mathbf{u}$ and that no numerical variable in $tree(p\,\mathrm{A}\,b\,\mathrm{L})$ is bound by the mentioned L. (Otherwise said: for no $q\,n \in^\wedge_\vee tree(p\,\mathrm{A}\,b\,\mathrm{L})$ we have that $n = \|q\|{+}1$.)

Then $\mathbf{u} \to_e \mathbf{u}[tree(p) := tree(p\,\mathrm{A})[tree(p\,\mathrm{A}\,b) := \{q\,n \in^\wedge_\vee tree(p\,\mathrm{A}\,b\,\mathrm{L})\}[upd]]$, where

$$upd \equiv n := n - 1\ \textit{if}\ n > \|q\|$$

5 Delayed updating

Another possibility is to make reduction easy, by not considering the updates of the numerical variables, until required. In this case we *delay all updates*. In the text below, we restrict this case to the *focused* balanced β-reduction described above (Section 4.2.3), but it can easily be extended to the more general balanced version of β-reduction discussed in Section 4.2.2.

There have been many proposals for describing the necessary updating in a formal way. The fist one has been described in de Bruijn, 1978a: an update function θ is added to a term constructor φ in order to update the num-labels. See also, e.g., Ventura *et al.*, 2015.

5.1 Focused beta-reduction with delayed updating

We use the symbol '$\rightarrow_{df}$' for the delayed, focused β-reduction.

Definition 5.1. Let $\mathbf{u} \in \mathcal{T}^{fre}$, let $b \in \mathbf{u}$ be a balanced path, assume that $p\,\mathrm{A}\,b\,\mathrm{L} \in^{\wedge} \mathbf{u}$ and that $q\,n$ is a fixed, complete path in $tree(p\,\mathrm{A}\,b\,\mathrm{L})$, where n is bound by L.

Then $\mathbf{u} \rightarrow_{df} \mathbf{u}\,[tree(p\,\mathrm{A}\,b\,\mathrm{L}) := tree(p\,\mathrm{A}\,b\,\mathrm{L})[q\,n := q\,n\,tree(p\,\mathrm{S})]]$.

Note that the edge labelled n stays where it is, and $tree(p\,\mathrm{S})$ is simply attached to it in $\rightarrow_{df}$-reduction, whereas this edge is *replaced* by an updated $tree(p\,\mathrm{S})$ in the original focused reduction. The remaining presence of the label n is to enable updating at a later stage.

For a pictorial representation, see Figure 4. Note: if $tree(p\,\mathrm{S})$ consists of a single edge only, labelled with a num-variable, then this edge is just attached to the open end of the edge labelled n, see Section 5.4 for an example.

The above definition implies that we have to revise our definition of λ-trees, since *var-cells now have open ends*: they may have connections to other cells at their bottom end. So num-variables no longer need to be end-labels in a path.

Definition 5.2. (*i*) Num-variables not being end-labels, we call *inner num-labels*.

(*ii*) To distinguish them from the (still present) num-labels that *are* end-labels, we call the latter *outer* num-labels (or leafs).

(*iii*) A λ-tree in which inner variables are allowed, we call an *extended* λ-tree.

Consequently, other definitions should be extended, as well. For example, the definition of a *balanced path* (Definition 3.20) must be adapted such that it allows inner num-labels *inside* the string of A's and L's.

In the remainder of this chapter, we assume that these definition revisions have been made. Moreover, when speaking about λ-trees without further restrictions, we mean extended *ones.*

Definition 5.3. The symbol $\mathcal{T}^{fre}$ concerns the set of name-free, *closed* trees *without* inner variables. The set of these trees where also inner variables are permitted, we denote by $\mathcal{T}^{fre+}$.

Hence, we must read $\mathbf{u} \in \mathcal{T}^{fre+}$ for $\mathbf{u} \in \mathcal{T}^{fre}$, in Definition 5.1.

Obviously, $\mathcal{T}^{fre} \subseteq \mathcal{T}^{fre+}$.

We define what *inclusion* of (extended) λ-trees means.

Definition 5.4. Let $\mathbf{u}$ and $\mathbf{u}'$ be λ-trees. Then $\mathbf{u} \subseteq \mathbf{u}'$ iff $p \in^{\wedge}_{\vee} \mathbf{u} \Rightarrow p \in^{\wedge} \mathbf{u}'$.

In Definition 5.1, right hand side, the copy of $tree(p\,\mathrm{S})$ attached to the edge labelled n, is not updated. Moreover, the complete tree $\mathbf{u}$ (including the edge labelled n) remains intact as integral part of $\mathbf{u}'$. Hence, we have the following theorem.

Theorem 5.5. *Let* $\mathbf{u}, \mathbf{u}' \in \mathcal{T}^{fre+}$. *Then* $\mathbf{u} \rightarrow_{df} \mathbf{u}'$ *implies* $\mathbf{u} \subset \mathbf{u}'$.

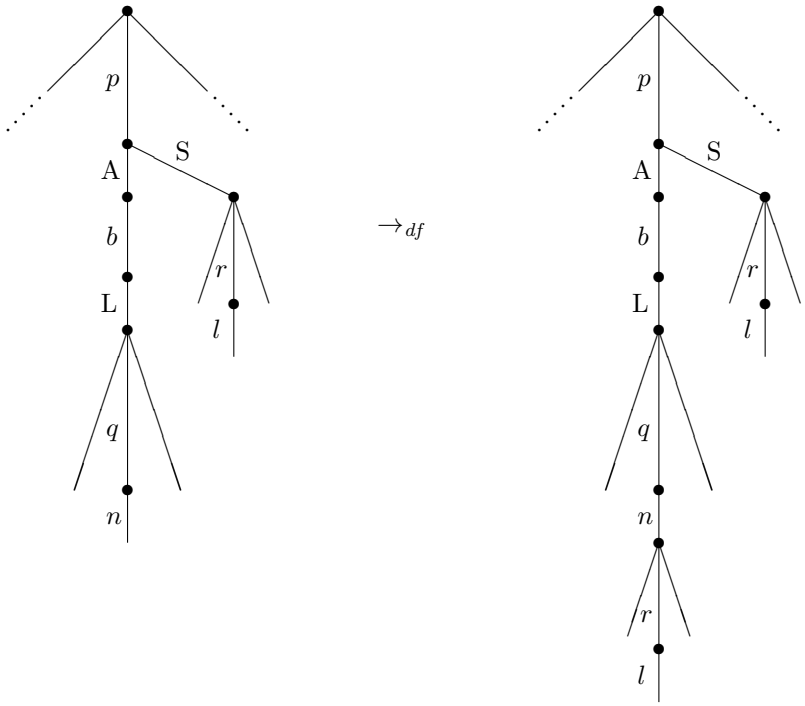

Figure 4: A picture of name-free β-reduction with delayed updating

The latter fact is clearly an advantage. But it comes with a price: when we wish to establish the relation between a leaf-variable of the copied $tree(p\,\mathrm{S})$ and its binder, we have to do more work. We discuss this in the following subsection.

5.2 Tracing the binder in reduction with delayed updating

Let $\mathbf{u}_0 \in \mathcal{T}^{fre}$ and $\mathbf{u}_0 \twoheadrightarrow_{df} \mathbf{u}$, so $\mathbf{u} \in \mathcal{T}^{fre+}$ is the result of a series of delayed, focused reductions. These reductions may introduce inner variables, so it is not immediately clear what the binders are for (inner or outer) variables. In this section we investigate how one can determine the binder of a variable in $\mathbf{u}$.

Let $p\,n \in^{\wedge} \mathbf{u}$, so $p\,n$ is a root path. Here n can be an inner or an outer num-label. We describe a pushdown automaton $\mathcal{P}_{fre}$ to find:

(i) the **L-binder** of n, i.e., the label $\mathrm{L} \in p$ that binds n (this label always exists, since $\mathcal{T}^{fre}$ only contains closed terms),

(ii) or the **A-binder** of n, i.e., the label $\mathrm{A} \in p$ that *matches* the L-binder of n, *if such an* A *exists* (this needs not to be the case).

The **use of the algorithm** is as follows. Let $\mathbf{u} \in \mathcal{T}^{fre+}$ and $p\,n \in^{\wedge} \mathbf{u}$. Assume that we desire to apply algorithm $\mathcal{P}_{fre}$ to find the L-binder or the A-binder of n.

In algorithm $\mathcal{P}_{fre}$, we employ *states* that are pairs of natural numbers: (k, l). We start with the insertion of a pair (n, i) in the tail of the string $p\,n$, *between* p and n. The automaton moves the pair *to the left* through p, one step at a time, successively passing the labels in p and meanwhile adapting the numbers in the pair. The automaton stops when the desired label (either the binding L or the A matching that L) has been found.

The automaton has an outside *stack* that will contain certain states that are *pushed* on the top of the stack; a state on top of the stack can also be *popped back*, i.e., inserted into the path p, again.

The *transitions* are described in Definition 5.6. A possible one-step transition is denoted by the symbol $\rightarrow$ (the reflexive, transitive closure of this relation is denoted $\twoheadrightarrow$). The procedure may be complicated by several recursive calls (see Example 5.10).

The *action* of $\mathcal{P}_{fre}$ is described in Note 5.9.

The formal description of $\mathcal{P}_{fre}$ is the following.

Preparation: Transform $p\,n$ into $p\,(n, i)\,n$, where i is 0 to find the L-binder, and 1 for the A-binder.

Now **start** $\mathcal{P}_{fre}$ employing the transition rules of Definition 5.6.

Definition 5.6. The algorithm $\mathcal{P}_{fre}$ is specified by the following rules:

(1) *first step: stack* $= \emptyset$
(2) p L (m, k) $q \;\rightarrow\; p\ (m{-}1, k)$ L q, if $m > 0$
(3) p A (m, k) $q \;\rightarrow\; p\ (m, k)$ A q, if $m > 0$
(4) p S (m, k) $q \;\rightarrow\; p\ (m, k)$ S q, if $m > 0$
(5) $p\ j\ (m, k)$ $q \;\rightarrow\; p\ (j, 1)\ j\ q$, if $m > 0$; *push* (m, k)
(6) p L $(0, l)$ $q \;\rightarrow\; p\ (0, l{+}1)$ L q, if $l > 0$
(7) p A $(0, l)$ $q \;\rightarrow\; p\ (0, l{-}1)$ A q, if $l > 0$
(8) $p\ j\ (0, l)$ $q \;\rightarrow\; p\ (0, l)\ j\ q$, if $l > 0$
(9*a*) $p\ (0, 0)$ $q \;\rightarrow\; p$ *pop* q, if *stack* $\neq \emptyset$
(9*b*) $p\ (0, 0)$ $q \;\rightarrow$ *stop*, if *stack* $= \emptyset$.

Definition 5.7. Let $\mathbf{u} \in \mathcal{T}^{fre+}$ and $p\,n \in^{\wedge} \mathbf{u}$.

(*i*) If $\mathcal{P}_{fre}$ is applied to $p\,(n, 0)\,n$ and stops in $p'\,(0, 0)\,q\,n$, then $q\,n$ is called the L-*block* of n.

(*ii*) If $\mathcal{P}_{fre}$ is applied to $p\,(n, 1)\,n$ and stops in $p'\,(0, 0)\,q\,n$, then $q\,n$ is called the A-*block* of n.

Lemma 5.8. (*i*)*: If* $q\,n$ *is the* L*-block of* n*, then* $q\,n \equiv \mathrm{L}\,q'\,n$ *and the mentioned* L *is the* L*-binder of* n*.*

(*ii*)*: If* $q\,n$ *is the* A*-block of* n*, then* $q\,n \equiv \mathrm{A}\,q'\,n$ *and the mentioned* A *is the* A*-binder of* n*.*

After the *preparation* and *step (1)*, the procedure $\mathcal{P}_{fre}$ has two possibly recursive *rounds*, **I** and **II**, with different actions. We explain this below.

Note 5.9. (*i*)*: Round* **I** *(steps (2) to (4)), gives a count-down of the* L*-labels in the path leftwards from an (inner or outer) num-label* n*; in this round,* A*-labels*

and S*-labels have no effect. When the count-down results in* $m = 0$*, then we just passed the* L *binding the original* n*. So we found the* L*-block of* n*. If the stack is empty, so* $\mathcal{P}_{fre}$ *is not in a recursion, then the* L*-block of the original variable has been found (step (9b)).*

(ii): When an inner variable j *is met in Round* **I** *(step (5)), then* $\mathcal{P}_{fre}$ *starts a recursive round, in which the* A*-block of that* j *is determined. The count-down of Round* **I** *is temporarily stopped and resumes immediately left of that* A*-block (step (9a)).* v5. 5–1

(iii) The A*-block of an inner variable* j *is found by executing Round* **I***, giving the* L*-block of* j*, immediately followed by Round* **II** *(steps (6) to (8)) determining the* r*-block connected with* j*. The latter is done by counting upwards for* L*'s and downwards for* A*'s. The* L*-block and the* r*-block combine in the desired* A*-block of* j*.*

(iv) Inner variables met in the search of an r*-block, are skipped (step (8)), because inner variables do not affect this Round* **II***-search. So no recursion is started inside an* r*-block.* v5. 5–2

(v) When determining an r*-block, no label* S *may be encountered, as can be seen in steps (6) to (8), where* S *is missing. (See also Definition 3.20.) This is due to the fact that for each* A*, a possibly matching* L *is positioned on the same 'branch', and not on a 'sub-branch'.*

5.3 An example of the action of algorithm $\mathcal{P}_{fre}$

We give an example of the execution of algorithm $\mathcal{P}_{fre}$.

Example 5.10. Firstly, we look for the L-binder of the final num-label – i.e., 3 – in the path A A L L A A L L L A A L A L L L A 2 S L 4 L L S 3.

So we take $i = 0$, and start $\mathcal{P}_{fre}$:

A A L L A A L L L A A L A L L L A 2 S L 4 L L S $(\mathbf{3},\mathbf{0})$ 3 (*stack* $= \emptyset$) →
A A L L A A L L L A A L A L L L A 2 S L 4 L L $(\mathbf{3},\mathbf{0})$ S 3 →
A A L L A A L L L A A L A L L L A 2 S L 4 L $(\mathbf{2},\mathbf{0})$ L S 3 →
A A L L A A L L L A A L A L L L A 2 S L 4 $(\mathbf{1},\mathbf{0})$ L L S 3 (*push* $(\mathbf{1},\mathbf{0})$) →
A A L L A A L L L A A L A L L L A 2 S L $(\mathbf{4},\mathbf{1})$ 4 L L S 3 →
A A L L A A L L L A A L A L L L A 2 S $(\mathbf{3},\mathbf{1})$ L 4 L L S 3 →
A A L L A A L L L A A L A L L L A 2 $(\mathbf{3},\mathbf{1})$ S L 4 L L S 3 (*push* $(\mathbf{3},\mathbf{1})$) →
A A L L A A L L L A A L A L L L A $(\mathbf{2},\mathbf{1})$ 2 S L 4 L L S 3 →
A A L L A A L L L A A L A L L L $(\mathbf{2},\mathbf{1})$ A 2 S L 4 L L S 3 →
A A L L A A L L L A A L A L L $(\mathbf{1},\mathbf{1})$ L A 2 S L 4 L L S 3 →
A A L L A A L L L A A L A L $(\mathbf{0},\mathbf{1})$ $\underbrace{\text{L L A 2}}_{\text{L-block of 2}}$ S L 4 L L S 3 →
A A L L A A L L L A A L A $(\mathbf{0},\mathbf{2})$ L L L A 2 S L 4 L L S 3 →
A A L L A A L L L A A L $(\mathbf{0},\mathbf{1})$ A L L L A 2 S L 4 L L S 3 →
A A L L A A L L L A A $(\mathbf{0},\mathbf{2})$ L A L L L A 2 S L 4 L L S 3 →
A A L L A A L L L A $(\mathbf{0},\mathbf{1})$ A L A L L L A 2 S L 4 L L S 3 →
A A L L A A L L L $(\mathbf{0},\mathbf{0})$ $\underbrace{\text{A A L A L L L A 2}}_{\text{A-block of 2}}$ S L 4 L L S 3 (*pop* $(\mathbf{3},\mathbf{1})$) →

$$\mathrm{A\,A\,L\,L\,A\,A\,L\,L\,L}\,(\mathbf{3},\mathbf{1})\,\mathrm{A\,A\,L\,A\,L\,L\,L\,A\,2\,S\,L\,4\,L\,L\,S\,3}\rightarrow$$
$$\mathrm{A\,A\,L\,L\,A\,A\,L\,L}\,(\mathbf{2},\mathbf{1})\,\mathrm{L\,A\,A\,L\,A\,L\,L\,L\,A\,2\,S\,L\,4\,L\,L\,S\,3}\rightarrow$$
$$\mathrm{A\,A\,L\,L\,A\,A\,L}\,(\mathbf{1},\mathbf{1})\,\mathrm{L\,L\,A\,A\,L\,A\,L\,L\,L\,A\,2\,S\,L\,4\,L\,L\,S\,3}\rightarrow$$
$$\mathrm{A\,A\,L\,L\,A\,A}\,(\mathbf{0},\mathbf{1})\,\underbrace{\mathrm{L\,L\,L\,A\,A\,L\,A\,L\,L\,L\,A\,2\,S\,L\,4}}_{\text{L-block of 4}}\mathrm{L\,L\,S\,3}\rightarrow$$
$$\mathrm{A\,A\,L\,L\,A}\,(\mathbf{0},\mathbf{0})\,\underbrace{\mathrm{A\,L\,L\,L\,A\,A\,L\,A\,L\,L\,L\,A\,2\,S\,L\,4}}_{\text{A-block of 4}}\mathrm{L\,L\,S\,3}\;\;(\mathit{pop}\;(\mathbf{1},\mathbf{0}))\rightarrow$$
$$\mathrm{A\,A\,L\,L\,A}\,(\mathbf{1},\mathbf{0})\,\mathrm{A\,L\,L\,L\,A\,A\,L\,A\,L\,L\,L\,A\,2\,S\,L\,4\,L\,L\,S\,3}\rightarrow$$
$$\mathrm{A\,A\,L\,L}\,(\mathbf{1},\mathbf{0})\,\mathrm{A\,A\,L\,L\,L\,A\,A\,L\,A\,L\,L\,L\,A\,2\,S\,L\,4\,L\,L\,S\,3}\rightarrow$$
$$\mathrm{A\,A\,L}\,(\mathbf{0},\mathbf{0})\,\underbrace{\mathrm{L\,A\,A\,L\,L\,L\,A\,A\,L\,A\,L\,L\,L\,A\,2\,S\,L\,4\,L\,L\,S\,3}}_{\text{L-block of 3}}\;(\mathit{stack}=\emptyset,\text{ hence }\mathit{stop})$$

Example 5.11. (*i*) In the case of Example 5.10, we can derive the A-block of the final num-label 3 by starting with $i = 1$. The derivation then needs three more steps.

(*ii*) Notice that several L-blocks and A-blocks arise during the execution of the procedure $\mathcal{P}_{fre}$, and that these can, on their own, be derived by means of a sub-calculation of the one above.

The procedure stops when the application of $\mathcal{P}_{fre}$ to a path $p\,n \in^{\wedge} \mathbf{u}$ results in the conclusion that a certain sub-path $\mathrm{L}\,q\,n$ (ending in the mentioned n) is an L-block. Then we have found that the displayed L binds the final n.

Lemma 5.12. *Let* $\mathbf{u} \in \mathcal{T}^{fre+}$ *and assume that* $\mathcal{P}_{fre}$ *has been applied to* $p\,n \in^{\wedge} \mathbf{u}$ *with conclusion that* $\mathrm{L}\,q\,n$ *is an* L*-block.*

Then each inner variable of q *is the end-variable of exactly one generated* A*-block and exactly one – corresponding – generated* L*-block.*

5.4 An example of beta-reduction with delayed updating

We consider the following untyped λ-term (in the usual notation) and three of its delayed, focused β-reducts. The underlining is meant to mark the redex parts, the dot above the variable (e.g., $\dot{y}$) marks the focus of the reduction. Transitions (*i*) $\rightarrow_f$ (*ii*) and (*ii*) $\rightarrow_f$ (*iii*) are one step reductions, (*iii*) $\twoheadrightarrow_f$ (*iv*) is two-step.

(*i*) $((\lambda x\,.\,(\underline{\lambda y}\,.\,\lambda z\,.\,\dot{y}\,z)\underline{\lambda u\,.\,x})x)\lambda v\,.\,v\;\rightarrow_f$

(*ii*) $((\lambda x\,.\,(\lambda y\,.\,\underline{\lambda z}\,.\,(\lambda u\,.\,x)\dot{z})\lambda u\,.\,x)\underline{x})\lambda v\,.\,v\;\rightarrow_f$

(*iii*) $((\underline{\lambda x}\,.\,(\lambda y\,.\,\lambda z\,.\,(\lambda u\,.\,x)\dot{x})\lambda u\,.\,\dot{x})x)\underline{\lambda v\,.\,v}\;\twoheadrightarrow_f$

(*iv*) $((\lambda x\,.\,(\lambda y\,.\,\lambda z\,.\,(\lambda u\,.\,x)\lambda v\,.\,v)\lambda u\,.\,\lambda v\,.\,v)x)\lambda v\,.\,v.$

In Figure 5.4 we represent these four λ-terms as trees. We mark the A-L-couples and the arguments in boxes. In the picture, we use a small triangle to mark the focus(ses).

Note 5.13. *Figure 5,* (*iii*)*, shows that it may occur that two (or more) num-labels occur adjacently. See the right-most part of the figure, at the end of the*

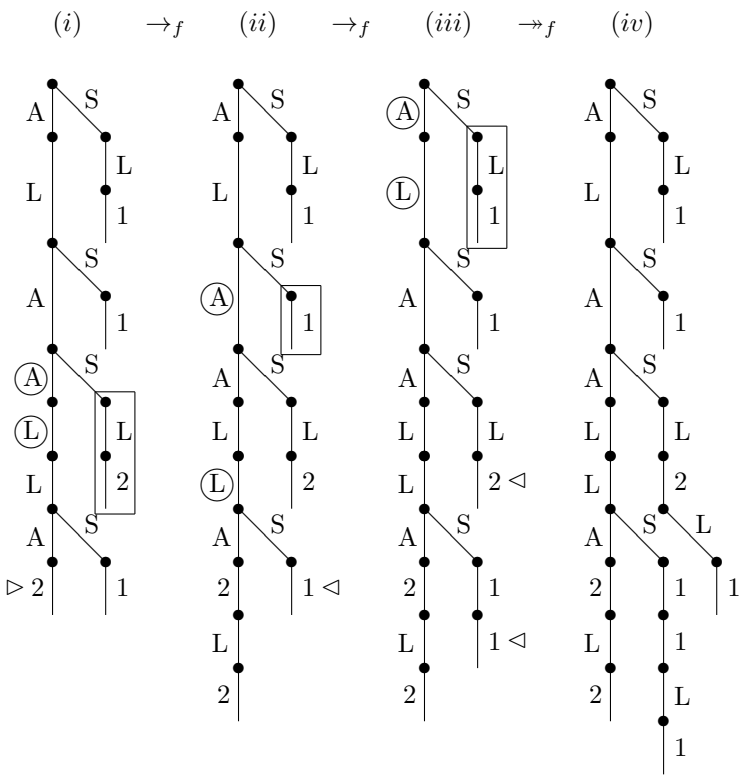

Figure 5: Examples of delayed, focused β-reduction

complete path A L A A L L S 1 1. *The first (or top-) label* 1 *is an inner numvariable, the other* 1 *is outer. Since the* A*-block of the first* 1 *is* A A L L S 1, *the* L*-block of the second* 1 *is* L A A L L S 1 1, *so the second* 1 *is bound to the leftmost, uppermost* L.

In Figure 5, (iv), both mentioned labels have become inner ones.

5.5 Name-free lambda-terms, without and with inner variables

In this section we compare $\mathcal{T}^{fre}$ with $\mathcal{T}^{fre+}$ and define mappings between them.

We only consider *closed* λ-trees, i.e., λ-trees in which all variables have been bound. We only consider λ-abstraction, so there are no P_x's or P's in the sets of terms studied below. (Hence, we limit ourselves to the set $\lambda_{\underline{\omega}}$ in Barendregt's cube; cf. Barendregt, 1992.) As to the mappings to be described, Π-abstraction behaves similarly to λ-abstraction.

Note 5.14. *We do not consider the mappings between* $\mathcal{T}^{car}$ *and* $\mathcal{T}^{fre}$ *and back,*

because these mapping have been described amply in the literature.

v6. 5–1

Definition 5.15. (*i*) Let $\mathbf{u_0} \in \mathcal{T}^{fre}$. The *structure* of $\mathbf{u_0}$, or $str(\mathbf{u_0})$, is $\mathbf{u_0}$ in which all variables have been omitted.

(*ii*) Let $\mathbf{u} \in \mathcal{T}^{fre+}$. The *structure* of $\mathbf{u}$, or $str(\mathbf{u})$, is (again) $\mathbf{u}$ in which all num-variables have been omitted. This concerns both the outer and the inner variables.

Moreover, in $str(\mathbf{u})$ all edges which did belong to inner labels have been erased (i.e., every path $p\,m\,q \in^{\wedge}_{\vee} \mathbf{u}$, where $q \neq \epsilon$, has been replaced by $p\,q$; this may, of course, need more than one round).

(*iii*) Let p be a path in $\mathbf{u} \in \mathcal{T}^{fre+}$. Then the *structure* of p, or $str(p)$, is p in which all (inner and outer) num-labels have been omitted.

Note that, in both cases, the final edges (the leaves) remain in the structures, albeit without num-labels.

The structure of a path obviously is a list of non-num-labels only, so elements of the set $\{\mathrm{L}, \mathrm{A}, \mathrm{S}\}$.

We describe a mapping from $\mathcal{T}^{fre+}$ to $\mathcal{T}^{fre}$. This mapping consists of two stages.

Let $\mathbf{u}$ be a λ-tree in $\mathcal{T}^{fre+}$.

(*i*) *Update* all outer num-labels as follows. Let $p\,k$ be a complete path in $\mathbf{u}$. Use the procedure $\mathcal{P}_{fre}$ to find the L in p that binds the final k. So $p\,k \equiv p_1\,\mathrm{L}\,p_2\,k$, where the displayed L binds the k. Then replace k by the number $\|p_2\|+1$.

(*ii*) *Erase* all inner num-labels and their edges. (So the tree changes, it becomes more 'compact'.)

Definition 5.16. Let $\mathbf{u} \in \mathcal{T}^{fre+}$. The combined procedure (*i*), followed by (*ii*), applied to $\mathbf{u}$, gives a mapping from $\mathcal{T}^{fre+}$ to $\mathcal{T}^{fre}$. We denote the resulting tree as $\overline{\mathbf{u}}$.

5.6 The behaviour of name-free lambda-trees under beta-reduction

In this section, we compare delayed, focused reduction on λ-terms in $\mathcal{T}^{fre+}$ with focused reduction for $\mathcal{T}^{fre}$. We choose to study *focused* β-reduction only, since the behaviour of any non-focused β-reduction can be described in terms of the focused one. v5. 5–3

We start with a general lemma, useful for any form of β-reduction. **v6. 5–2**

Lemma 5.17. (*i*) *Let* $\mathbf{u} \in \mathcal{T}^{fre+}$ *and* $c \equiv p\,\mathrm{S}\,r\,n \in^{\wedge}_{\vee} \mathbf{u}$. *Then the leaf* n *is bound by a certain* L *in either* p *or* r. *I.e.,* $p \equiv p_1\,\mathrm{L}\,p_2$ *or* $r \equiv r_1\,\mathrm{L}\,r_2$ *for this* L.

(*ii*) *Let* a *be any* A*-block. Let* c' *be* c *of* (*i*) *in which* S *has been replaced by* a. *Then leaf* n *is bound by an* L *in either* p *or* r *at a position corresponding to the one in* (*i*).

Proof If $\mathrm{L} \in r$, then trivial.

Assume $\mathrm{L} \in p$, say $p \equiv p_1 \,\mathrm{L}\, p_2$. Apply algorithm $\mathcal{P}_{fre}$:

(i) $p\,\mathrm{S}\,r\,(n,0)\,n \twoheadrightarrow^{(1)} p\,\mathrm{S}\,(m,0)\,r\,n \twoheadrightarrow p\,(m,0)\,\mathrm{S}\,r\,n \equiv p_1\,\mathrm{L}\,p_2\,(m,0)\,\mathrm{S}\,r\,n \twoheadrightarrow^{(2)} p_1\,(0,0)\,\mathrm{L}\,p_2\,\mathrm{S}\,r\,n$.

(ii) Let $a \equiv a'\,k$. Then: $p\,a\,r\,(n,0)\,n \twoheadrightarrow^{\mathrm{see}\,(1)} p\,a'\,k\,(m,0)\,r\,n \twoheadrightarrow^{\mathrm{push}\,(m,0)} p\,a'\,(k,1)\,k\,r\,n \twoheadrightarrow^{\mathrm{Def.\ 5.7}(ii)} p\,(0,0)\,a'\,k\,r\,n \twoheadrightarrow^{\mathrm{pop}\,(m,0)} p\,(m,0)\,a\,r\,n \equiv p_1\,\mathrm{L}\,p_2\,(m,0)\,a\,r\,n \twoheadrightarrow^{\mathrm{see}\,(2)} p_1\,(0,0)\,\mathrm{L}\,p_2\,a\,r\,n$.

So $\mathrm{L}\,p_2\,\mathrm{S}\,r\,n$ and $\mathrm{L}\,p_2\,a\,r\,n$ are L-blocks by Definition 5.7(i), and n is bound by an L at corresponding positions in both cases. □

Note 5.18. *At transition* (1) *in* (i)*, the reached state must be* $(m,0)$ *for some* m*, and not* (m,l) *for some* $l > 0$*, since the latter case only occurs when the displayed* S *is part of an* r*-block. This would contradict what we said in Note 5.9,* (v)*.*

It will be necessary that we can refer to redexes in $\mathcal{T}^{fre+}$ (so also in $\mathcal{T}^{fre}$, since $\mathcal{T}^{fre} \subset \mathcal{T}^{fre+}$).

Definition 5.19. (i) Let $c \equiv p\,\mathrm{A}\,b\,\mathrm{L}\,r\,k \in^{\wedge}_{\vee} \mathbf{u_0} \in^{\wedge}_{\vee} \mathcal{T}^{fre+}$, with balanced path b, such that the displayed L binds the outer variable k (verifiable by $\mathcal{P}_{fre}$). Then p identifies a redex and the pair $\rho \equiv (p,r)$ identifies a *focused* reduction, outer variable k being the focus. We call this a (p,r)-reduction. (Cf. Definition 4.8.)

We write $\mathbf{u} \to^{\rho}_{pf} \mathbf{u}'$ for the focused β-reduction generated by ρ in $\mathcal{T}^{fre+}$. Note that $\to^{\rho}_{pf}$ acts as a *function*, i.e., for fixed $\mathbf{u}$ and fixed redex ρ there exists precisely one $\mathbf{u}'$.

(ii) Now look at $\mathcal{T}^{fre}$. Assume that $\mathbf{u_0} \in \mathcal{T}^{fre}$ has the same structure as $\mathbf{u} \in \mathcal{T}^{fre+}$, mentioned in (i). There exists a path $\overline{c} \equiv \overline{p}\,\mathrm{A}\,\overline{b}\,\mathrm{L}\,\overline{r}\,l \in^{\wedge}_{\vee} \mathbf{u_0}$ with the same structure as c, such that $str(p) \equiv str(\overline{p})$, and so on. Assume that l is the 'updated' k. Then the pair $\overline{\rho} \equiv (\overline{p},\overline{r})$ identifies a focused reduction in $\mathcal{T}^{fre}$, corresponding to ρ in $\mathcal{T}^{fre+}$.

We write $\overline{\mathbf{u}} \to^{\overline{\rho}}_{f} \overline{\mathbf{u}'}$ for the focused β-reduction in $\mathcal{T}^{fre}$, generated by $\overline{\rho}$. Again, $\to^{\overline{\rho}}_{f}$ acts as a function.

Now we are ready to formulate the main theorem, describing the connection between corresponding reductions in $\mathcal{T}^{fre+}$ and $\mathcal{T}^{fre}$.

Theorem 5.20. *Let* $\mathbf{u}_1 \in \mathcal{T}^{fre+}$*. Assume that* $\mathbf{u_1} \to^{\rho}_{df} \mathbf{u_2}$*. Then* $\overline{\mathbf{u_1}} \to^{\overline{\rho}}_{f} \overline{\mathbf{u_2}}$*.*

Proof It suffices to investigate what happens with

(1) the complete path $c_1 \equiv p\,\mathrm{A}\,b\,\mathrm{L}_0\,q\,n \in \mathbf{u}_1$, where n is the *focus* of the one-step $\to^{\rho}_{df}$-reduction, while n is bound by the displayed L_0. We abbreviate the path $\mathrm{A}\,b\,\mathrm{L}_0\,q$ in c_1, to a. So $\boxed{c_1 \equiv p\,a\,n}$. (Note that $a\,n$ is an A-block.)

(2) the 'argument' *tree*$(p\,\mathrm{S})$ of the reduction step $\mathbf{u}_1 \to^{\rho}_{df} \mathbf{u}_2$ in $\mathcal{T}^{fre+}$; to be precise, we prefer to consider the *grafted* tree $p\,\mathrm{S}\,$*tree*$(p\,\mathrm{S})$. Without loss of generality, we only consider *one (relatively complete)* path in that grafted tree, say $\boxed{c_2 \equiv p\,\mathrm{S}\,r\,m}$.

We use the name L' for the L binding m in $c_2 \equiv p\,\mathrm{S}\,r\,m$. Note that L' may occur in either r or p (cf., Lemma 5.17,(i)), so we distinguish the two *cases* $\bullet_r:\ r \equiv r_1\,\mathrm{L}'\,r_2$ and $\bullet_p:\ p \equiv p_1\,\mathrm{L}'\,p_2$.

Assume $\mathbf{u_1} \to^{\rho}_{df} \mathbf{u_2}$ and $\overline{\mathbf{u}_1} \to^{\overline{\rho}}_{f} \mathbf{v}$. We only have to prove now that $\mathbf{v} \equiv \overline{\mathbf{u}_2}$.

We investigate below how c_1 and c_2 behave in the cases (i) and (ii). We display this in four diagrams.

Note 5.21. *In the diagrams below, we deal loosely with the reduction symbols, assuming that the intention will be clear. In two places we add symbol $*$, meaning that we use Lemma 5.17,(ii). In both cases, we introduce variable m_{dup}, being a 'duplicate' of the m occurring in tree$(p\,S)$. Ths doing we distinguish the two m's, having different positions in the trees $\mathbf{u}_1$ and $\overline{\mathbf{u}}_2$, or $\overline{\mathbf{u}}_1$ and $\mathbf{v}$, respectively.*

Continuation of the proof

(i) $\mathbf{u}_1 \to^{\rho}_{df} \mathbf{u}_2 \Rightarrow \overline{\mathbf{u}_2}$.

$c_1 \equiv p\,a\,n \to^{\rho}_{df}{}^{*}\ p\,a\,n\,r\,m_{dup}\ \Rightarrow\ \overline{p}\,\overline{a}\,\overline{r}\,\overline{m_{dup}}$	$\overline{m_{dup}} \equiv \begin{cases} \bullet_r : \overline{m} \\ \bullet_p : \overline{m} + \lVert a \rVert \end{cases}$
$c_2 \equiv p\,S\,r\,m \to^{\rho}_{df} p\,S\,r\,m\ \Rightarrow\ \overline{p}\,S\,\overline{r}\,\overline{m}$	$\overline{m} \equiv \begin{cases} \bullet_r : \lVert r_2 \rVert{+}1 \\ \bullet_p : \lVert p_2\,r \rVert{+}1 \end{cases}$

(ii) $\mathbf{u}_1 \Rightarrow \overline{\mathbf{u}_1} \to^{\overline{\rho}}_{f} \mathbf{v}$.

$c_1 \equiv p\,a\,n \Rightarrow \overline{p}\,\overline{a}\,\overline{n} \to^{\overline{\rho}}_{f}{}^{*}\ \overline{p}\,\overline{a}\,\overline{r}\,\overline{m}_{dup}$	$\overline{n} \equiv \lVert q \rVert{+}1;\ \overline{m}_{dup} \equiv \begin{cases} \bullet_r : \overline{m} \\ \bullet_p : \overline{m} + \lVert a \rVert \end{cases}$
$c_2 \equiv p\,S\,r\,m \Rightarrow \overline{p}\,S\,\overline{r}\,\overline{m} \to^{\overline{\rho}}_{f} \overline{p}\,S\,\overline{r}\,\overline{m}$	$\overline{m} \equiv \begin{cases} \bullet_r : \lVert r_2 \rVert{+}1 \\ \bullet_p : \lVert p_2\,r \rVert{+}1 \end{cases}$

Now first of all, $str(\mathbf{v}) \equiv str(\overline{\mathbf{u}_2})$, as is easy to see. And since it turns out that $\overline{m_{dup}}$ in $\overline{\mathbf{u}_2}$ has the same position and the same value as $\overline{m}_{dup}$ in $\overline{\mathbf{v}}$ (and that $\overline{m}$'s with identical values occur in both $\overline{\mathbf{u}_2}$ and $\mathbf{v}$), we may conclude that *all* outer corresponding variables in $\overline{\mathbf{u}_2}$ and $\mathbf{v}$ are identical. Hence, $\overline{\mathbf{u}_2} \equiv \mathbf{v}$, as required. □

We continue with a number of direct consequences of Theorem 5.20.

In the following corollary we use the well-known correspondence between β-reductions in $\mathcal{T}^{car}$ and $\mathcal{T}^{fre}$.

Corollary 5.22. *Let $\mathbf{u_1} \in \mathcal{T}^{fre+}$ and assume that ρ identifies a focused β-reduction such that $\mathbf{u_1} \to^{\rho}_{df} \mathbf{u_2}$.*

Furthermore, let $\mathbf{t_1}$ and $\mathbf{t_2}$ be the name-carrying versions of $\overline{\mathbf{u_1}}$ and $\overline{\mathbf{u_2}}$ and let ρ' be the redex in $\mathbf{t_1}$ that corresponds to ρ in $\mathbf{u_1}$.

Then $\mathbf{t_1} \to^{\rho'}_{f} \mathbf{t_2}$. Moreover, for $i = 1, 2$, $str(\mathbf{t_i}) \equiv str(\mathbf{u_i})$.

Next, we compare a sequence of delayed, focused β-reductions in $\mathcal{T}^{fre+}$ with the corresponding sequence of focused β-reductions in $\mathcal{T}^{fre}$.

Corollary 5.23. *Let* $\mathbf{u}_1 \in \mathcal{T}^{fre+}$ *and assume* $\mathbf{u}_1 \to_{df}^{\rho_1} \mathbf{u}_2 \to_{df}^{\rho_2} \ldots \to_{df}^{\rho_{n-1}} \mathbf{u}_n$, *where* $\rho_1, \ldots, \rho_{n-1}$ *identify the chosen one-step focused reductions.*

Then there is a corresponding sequence of focused reductions in $\mathcal{T}^{fre}$*:* $\overline{\mathbf{u}_1} \to_f^{\overline{\rho_1}} \overline{\mathbf{u}_2} \to_f^{\overline{\rho_2}} \ldots \to_f^{\overline{\rho_{n-1}}} \overline{\mathbf{u}_n}$.

Moreover, $str(\mathbf{u}_i) \equiv str(\overline{\mathbf{u}_i})$ *for all* $1 \leq i \leq n$.

5.7 Theorems on delayed updating

We single out the λ-trees in $\mathcal{T}^{fre+}$ that 'originate' from $\mathcal{T}^{fre}$ and give a number of theorems and lemmas concerning these trees.

Definition 5.24. If $\mathbf{u} \in \mathcal{T}^{fre+}$ has the property that there is an $\mathbf{u_0} \in \mathcal{T}^{fre}$ with $\mathbf{u_0} \to_{df} \mathbf{u}$, then $\mathbf{u}$ is called *legal*. Such an $\mathbf{u_0}$ is called an *origin* of $\mathbf{u}$.

Lemma 5.25. *If* $\mathbf{u} \in \mathcal{T}^{fre+}$ *is legal, then its origin is unique.*

We continue with some lemma's concerning inner and outer variables. (See also Lemma 3.24.)

Lemma 5.26. *Let* $\mathbf{u} \in \mathcal{T}^{fre+}$ *be legal. Assume* $p \in^{\wedge}_{\vee} \mathbf{u}$ *such that* $p \equiv p_1\, n\, p_2$.

(*i*) *Let* p_2 *be non-empty (so* n *is an* inner *variable of* $\mathbf{u}$*). Then* n *is the final variable of a corresponding* A*-block enclosing a corresponding* L*-block, both being subpaths of* $p_1\, n$*. Moreover, the* A*-block and the* L*-block are corresponding, and the front-*L *of the* L*-block binds* n.

(*ii*) *Let* p_2 *be empty (so* n *is an* outer *variable of* $\mathbf{u}$*). Then* n *is the final variable of a corresponding* L*-block being a subpath of* $p_1\, n$*. Moreover, the front-*L *of the* L*-block binds* n*. (Note: there may also be a corresponding* A*-block, enclosing the* L*-block, but this is not necessarily so.)*

Next, we give a general theorem for reductions in $\mathcal{T}^{fre}$ and $\mathcal{T}^{fre+}$ (cf. Theorem 3.29).

Theorem 5.27. (*i*) *Each of the reductions* $\to_b$, $\to_f$ *and* $\to_e$ *is confluent for name-free paths.*

(*ii*) *The reduction* $\to_{df}$ *is confluent.*

Lemma 5.28. *Let* $\mathbf{u} \in \mathcal{T}^{fre+}$ *be legal. Let* $\mathbf{u_0}$ *be defined as the set of all paths* $p\, n \in^{\wedge} \mathbf{u}$ *such that* p *has no inner variables. Then* $\mathbf{u_0}$ *is the origin of* $\mathbf{u}$.

Lemma 5.29. *Let* $\mathbf{u}$ *be a legal* λ*-tree with origin* $\mathbf{u}_0$*. Then there is a* procedure *to find an* $\to_{df}$*-reduction sequence such that* $\mathbf{u_0} \twoheadrightarrow_{df} \mathbf{u}$.

Lemma 5.30. *Let* $\mathbf{u}$ *be a legal* λ*-tree and* $\mathbf{u} \twoheadrightarrow_{df} \mathbf{u}'$*. Assume that path* $p \in^{\wedge} \mathbf{u}$ *and that* $tree(p)$ *is a redex in* $\mathbf{u}$*, with argument* $tree(p\, S)$*. Then:*

(*i*) $tree(p)$ *is also a redex in* $\mathbf{u}'$*,*

(*ii*) *argument* $tree(p\, S)$ *in* $\mathbf{u}$ *is a subtree of argument* $tree(p\, S)$ *in* $\mathbf{u}'$.

We now discuss some important consequences of Theorem 5.5. **v6. 5–3**

Definition 5.31. Let $\mathbf{u} \in \mathcal{T}^{fre+}$ and $p \in^\wedge \mathbf{u}$. The *trail* of p, or *trail*(p), is the non-empty sequence of edges obtained from p by omitting all labels, but preserving the 'directions' of the edges; a *direction* being `l` ('left') or `r` ('right'). This is done such that labels A and L are reflected as `l` in *trail*(p) and label S as `r`. For the (unary) num-labels (inner and outer), we choose `l`.

Example 5.32. Let p be L L A 2 S L S 1 L, then *trail*$(p) \equiv$ `llllrlrll`.

The following lemma is a consequence of the construction principles for trees.

Lemma 5.33. *Let* $\mathbf{u} \in \mathcal{T}^{fre+}$ *and* $p_1, p_2 \in^\wedge \mathbf{u}$ *such that* $|p_1| = |p_2|$. *If* *trail*$(p_1) \equiv$ *trail*(p_2), *then* $p_1 \equiv p_2$. v5. 5–4

Proof First we show that, if $p_1 \equiv q\,\mathtt{l}_1\,\mathtt{q}'$ and $p_2 \equiv q\,\mathtt{l}_2\,\mathtt{q}''$, where $\mathtt{l}_1$ and $\mathtt{l}_2$ are labels, then $\mathtt{l}_1 \equiv \mathtt{l}_2$. This is done by distinguishing the different possibilities for $\mathtt{l}_1$ and using construction principles of λ-trees.

Next, apply induction on $|q|$. Since $|p_1| = |p_2|$, also $p_1 \equiv p_2$. □

Theorem 5.34. *Let* $\mathbf{u} \in \mathcal{T}^{fre}$ *and assume* $\mathbf{u} \twoheadrightarrow_{df} \mathbf{u_1}$ *and* $\mathbf{u} \twoheadrightarrow_{df} \mathbf{u_2}$. *Assume that* $p_1 \in^\wedge \mathbf{u_1}$ *and* $p_2 \in^\wedge \mathbf{u_2}$ *such that* *trail*$(p_1) \equiv$ *trail*(p_2). *Then* $p_1 \equiv p_2$.

Proof By 'confluence' (Theorem 5.27), there exists $\mathbf{u_3} \in \mathcal{T}^{fre+}$ such that $\mathbf{u_1} \twoheadrightarrow_{df} \mathbf{u_3}$ and $\mathbf{u_2} \twoheadrightarrow_{df} \mathbf{u_3}$. Then by Theorem 5.5, $\mathbf{u_1} \subset \mathbf{u_3}$ and $\mathbf{u_2} \subset \mathbf{u_3}$. Hence $p_1, p_2 \in^\wedge \mathbf{u_3}$, and by Lemma 5.33, $p_1 \equiv p_2$. □

Definition 5.35. (*i*) By $\mathcal{T}_{bin}$ we denote the infinite binary tree that is the graphic representation of the (infinite) set of all infinite lists composed of `l`'s and `r`'s.

(*ii*) Each trail `t` is a non-empty, finite initial part of some element of $\mathcal{T}_{bin}$. We denote this correspondence by $\mathtt{t} \in^\wedge \mathcal{T}_{bin}$.

(*iii*) An edge in $\mathcal{T}_{bin}$ is called a *position* in $\mathcal{T}_{bin}$.

Definition 5.36. (*i*) Let $\mathtt{t} \in^\wedge \mathcal{T}_{bin}$ and assume that the final element of `t` corresponds to position ϵ in $\mathcal{T}_{bin}$. Then we say that `t` *marks* ϵ.

(*ii*) Let $\mathbf{u} \in \mathcal{T}^{fre+}$ and assume that there is a non-empty $p \in^\wedge \mathbf{u}$ with final label ℓ, such that *trail*(p) marks position ϵ. Then we say that $\mathbf{u}$ *covers* position ϵ *with label* ℓ. Moreover, we say that position ϵ is *occupied* by label ℓ.

Now we can show that, given a λ-tree $\mathbf{u} \in \mathcal{T}^{fre}$ and some $\rightarrow_{df}$-reduct $\mathbf{u}'$ of $\mathbf{u}$ (i.e., a $\mathbf{u}' \in \mathcal{T}^{fre+}$ with $\mathbf{u} \twoheadrightarrow_{df} \mathbf{u}'$) that covers position ϵ with label ℓ, then this label is *unique* – whatever the $\mathbf{u}'$ is.

Theorem 5.37. *Let* $\mathbf{u} \in \mathcal{T}^{fre}$ *and let* ϵ *be a position in* $\mathcal{T}_{bin}$. *Then:*

– either *there is no* $\rightarrow_{df}$*-reduct of* $\mathbf{u}$ *that covers* ϵ,

– or *there is such an* $\mathbf{u}'$*; assume that this* $\mathbf{u}'$ *covers* ϵ *with label* ℓ*; then* all $\rightarrow_{df}$*-reducts of* $\mathbf{u}$ *covering* ϵ, *cover* ϵ *with the same label* ℓ.

Proof Use Theorem 5.34. □

We conclude with a theorem stating that weak normalization (WN) implies strong normalization (SN) in $\mathcal{T}^{fre+}$. We firstly give a definition.

v6. 5–4

Definition 5.38. Let $\mathbf{u} \in \mathcal{T}^{fre+}$. Then $\varphi(\mathbf{u})$ is the total number of inner and outer variable occurrences in $\mathbf{u}$.

Lemma 5.39. *Let* $\mathbf{u} \in \mathcal{T}^{fre+}$ *and* $\mathbf{u} \twoheadrightarrow_{df} \mathbf{u}'$. *Then* $\varphi(\mathbf{u}) < \varphi(\mathbf{u}')$.

Proof By Theorem 5.5: $\mathbf{u} \subset \mathbf{u}'$. □

v6. 5–5

Theorem 5.40. *Let* $\mathbf{u} \in \mathcal{T}^{fre+}$ *be weakly normalizing. Then* $\mathbf{u}$ *is also strongly normalizing.*

Proof Let $\mathbf{v}$ be a normal form of $\mathbf{u}$. Assume that there is an infinite reduction sequence $\mathbf{u} \to_{df} \mathbf{u_1} \to_{df} \mathbf{u_2} \to_{df} \ldots$. Then $\varphi(\mathbf{u}) < \varphi(\mathbf{u_1}) < \varphi(\mathbf{u_2}) < \ldots$ by Lemma 5.39.

By confluence (Theorem 5.27) and normality of $\mathbf{v}$, for all i: $\mathbf{u_i} \twoheadrightarrow_{df} \mathbf{v}$, so also for all i: $\varphi(\mathbf{u_i}) < \varphi(\mathbf{v})$. This easily leads to a contradiction. □

6 Concluding Remarks

v5. 6–1

In this article, we proposed for the first time the system $\mathcal{T}^{fre+}$ of λ-calculus featuring the simplest support for delaying α-conversion of function arguments in β-reduction. To that aim, our system needs an extension of β-reduction known as balanced or distant β-reduction. The system $\mathcal{T}^{fre+}$ features depth-indexed variable references (Section 5). **v6. 6–1**

A corresponding system for *name-carrying* variables can be defined straightforwardly. Moreover, it is not hard to define a procedure $\mathcal{P}_{car}$ for the name-carrying system, with similar properties as $\mathcal{P}_{fre}$ has for the name-free system.

As we saw, $\mathcal{T}^{fre+}$ is confluent and weak normalization implies strong normalization. Moreover, we can map each term of $\mathcal{T}^{fre+}$ to a term of traditional λK so that each β-reduction step in $\mathcal{T}^{fre+}$ corresponds to a β-reduction step in λK and vice versa. v5. 6–2

The first author, being experienced in computer-assisted proof verification, is checking the present theory of $\mathcal{T}^{fre+}$ with Matita (Asperti *et al.*, 2011; Guidi, 2014) and plans to report on this activity in future work. Moreover, we plan to relate $\mathcal{T}^{fre+}$ to other systems featuring delayed α-conversion like the ones we mentioned in Section 2.

References

Abadi, M., Cardelli, L., Curien, P.-L. and Lévy, J.-J., Explicit Substitutions, *J. of Functional Programming*, Vol. 1, 375–416, Cambridge University Press, 1991.

Accattoli, B. and Kesner, D., The structural lambda-calculus. In Dawar, A. and Veith, H. eds, *CSL 2010*, Vol. 6247 of LNCS, 381-–395, Springer, 2010.

Accattoli, B. and Kesner, D., The permutative lambda calculus, *18th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning - LPAR-18*, Merida, Venezuela, 2012

Asperti, A., Ricciotti, W., Sacerdoti Coen, C. and Tassi, E., The Matita Interactive Theorem Prover. In Bjørner, N. and Sofronie-Stokkermans, V., eds, *Proceedings of the 23rd International Conference on Automated Deduction (CADE-2011)*, Vol. 6803 of LNCS, 64–69, Springer, 2011.

Barenbaum, P. and Bonelli, E., Optimality and the Linear Substitution Calculus. In: Miller, D., ed., *2nd International Conference on Formal Structures for Computation and Deduction (FSCD 2017)*, 9:1—9:16, Leibniz International Proceedings in Informatics, 2017.

Barendregt, H.P., Lambda calculi with types. In Abramsky, S., Gabbay, D.M. and Maibaum, T., eds, *Handbook of Logic in Computer Science*, Vol. 2, 117–309, Oxford, 1992.

de Bruijn, N.G., Lambda calculus notation with nameless dummies, a tool for automatic formula manipulation, with application to the Church-Rosser theorem, *Indagationes Math.* 34 (1972), 381–392. Also in Nederpelt *et al.* (1994).

de Bruijn, N.G., *A namefree lambda calculus with formulas involving symbols that represent reference transforming mappings*, Eindhoven University of Technology, Dept. of Math., Memorandum 1977-10, 1977. (See also The Automath Archive AUT 050, www.win.tue.nl>Automath.) v5. Ref–1

de Bruijn, N.G. (1978a), *A namefree lambda calculus with facilities for internal definitions of expressions and segments*, Eindhoven University of Technology, EUT-report 78-WSK-03, 1978. (See also The Automath Archive AUT 059, www.win.tue.nl>Automath.) v5. Ref–2

de Bruijn, N.G. (1978b), Lambda calculus notation with namefree formulas involving symbols that represent reference transforming mappings, *Indagationes Math.* 81, 348–356, 1978. (See also The Automath Archive AUT 055, www.win.tue.nl>Automath.)

Curien, P.-L., Hardin, Th. and Lévy, J.-J., Confluence Properties of Weak and Strong Calculi of Explicit Substitutions, *Journal of the ACM*, 43(2), 362–397, New York, 1996.

Guidi, F., *Landau's "Grundlagen der Analysis" from Automath to lambda-delta*, University of Bologna, Technical Report UBLCS 2009-16, 2009.

Guidi, F., *Formal specification for the interactive prover Matita 0.99.2*, 2014. (`http://lambdadelta.info/`)

Kamareddine, F.D. and Nederpelt, R.P., On stepwise explicit substitution, *Int. J. of Foundations of Computer Science*, 4(3), 197–240, World Scientific Publishing Co, Singapore, 1993.

Kluge, W., Abstract Computing Machines — A Lambda Calculus Perspective, *Texts in Theoretical Computer Science*, EATCS Series, Springer-Verlag, 2005.

Nederpelt, R.P., *Strong normalisation in a typed lambda-calculus with lambda-structured types*. Ph.D. thesis, Eindhoven University of Technology, 1973. Also in Nederpelt *et al.* (1994).

Nederpelt, R.P., *A system of lambda-calculus possessing facilities for typing and abbreviating, Part I: Informal introduction*, Memorandum 1979-02, Department of Mathematics, Eindhoven University of Technology, The Automath Archive AUT 068, www.win.tue.nl>Automath, 1979.

Nederpelt, R.P., *A system of lambda-calculus possessing facilities for typing and abbreviating, Part II: Formal description*, Memorandum 1980-11, Department of Mathematics, Eindhoven University of Technology, The Automath Archive AUT 075, www.win.tue.nl>Automath, 1980.

Nederpelt, R.P., Geuvers, J.H. and de Vrijer, R.C., eds: *Selected Papers on Automath*, North-Holland, Elsevier, 1994.

Ventura, D. L., Kamareddine, F. and Ayala-Rincon, M., Explicit substitution calculi with de Bruijn indices and intersection type systems, *The Logic Journal of the Interest Group of Pure and Applied Logic*, Vol. 23, issue 2, 295–340, Oxford, 2015.

v6. bib