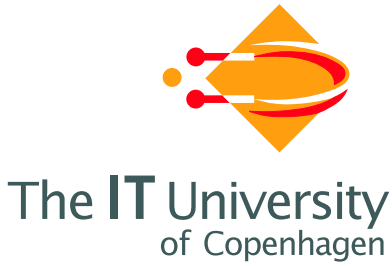


Λ_{sub} as an explicit substitution calculus

Shane O'Conchúir

This work was partially supported by funding from the Irish Research Council for Science, Engineering and Technology: funded by the National Development Plan.

IT University Technical Report Series **TR-2006-95**

ISSN 1600–6100 **9 2006**

ISSN 1600–6100

ISBN 87-7949-139-1

Copies may be obtained by contacting:

IT University of Copenhagen
Rued Langgaards Vej 7,
DK-2300 København S
Denmark

Telephone: **+45 72 18 50 00**
Telefax: **+45 72 18 50 01**
Web `www.itu.dk`

Λ_{sub} as an explicit substitution calculus

Shane O'Conchúir

Abstract

This work explores confluence and termination in Milner's encoding of the λ-calculus as a bigraphical reactive system. In that work, the λ-calculus was extended with explicit subsitutions and the extension (Λ_{sub}) was encoded as a bigraphical reactive system.

We prove that the reduction relation of the extension is confluent on ground terms and preserves strong normalisation (PSN) of β-reduction. This gives us corresponding proofs for the bigraphical encoding. The proofs are based on the strong relationship between Λ_{sub} and the calculus $\lambda\mathrm{xgc}$ of Bloo and Rose. The notion of composition of substitutions in Λ_{sub} and the problems it raises when attempting to prove PSN are discussed.

We then exploit similarities between Λ_{sub} and the $\lambda\mathrm{lxr}$ calculus of Kesner and Lengrand to present a translation from Λ_{sub} to a modified version of $\lambda\mathrm{lxr}$. We show that reduction in the former may be simulated in the latter which leads to a clearer proof of PSN for Λ_{sub}.

Contents

Introduction

This report concerns Λ_{sub}, a λ-calculus with explicit substitutions inspired by the $\lambda\sigma$ calculus of Abadi et al. [ACCL91] and presented by Milner [Mil04][1]. The calculus has been used to present an encoding, $'\Lambda_{\mathrm{BIG}}$, of the λ-calculus in the bigraphical framework of Milner, Leifer, and Jensen [Mil01, Lei01, JM04, Mil05b] but we concentrate here on the properties that Λ_{sub} has when viewed as an explicit substitution calculus.

Section 1.1 introduces the explicit substitution calculus Λ_{sub}. We describe aspects of the bigraphical encoding and compare the encoding and its reduction strategy to previous work in the π-calculus [Mil90] and the fusion calculus [PV98]. We also show that $'\Lambda_{\mathrm{BIG}}$ encodes the full reduction strategy for the λ-calculus.

Section 1.2 describes the calculus $\lambda\mathrm{xgc}$ of Bloo and Rose [BR95] and compares it to Λ_{sub}. Although the calculi were developed independently and Λ_{sub} was based on $\lambda\sigma$, we find that Milner's calculus more closely resembles $\lambda\mathrm{xgc}$. The main difference as we will see is in how substitutions are performed.

The similarities between Λ_{sub} and $\lambda\mathrm{xgc}$ led us to ask whether the proofs of confluence and preservation of strong normalisation (PSN) for $\lambda\mathrm{xgc}$ could be applied to Λ_{sub}. We investigate this in Sections 2.1 and 2.2. In Section 2.1, we prove that the rewrite relation of Λ_{sub} is confluent. Section 2.2 contains proofs that certain subsets of the rewrite relation preserve strong normalisation of β-reduction. The proofs rest heavily on previous work by Bloo and Rose [BR95] where these two properties were proved for $\lambda\mathrm{xgc}$, a calculus inspired by $\lambda\sigma$. We use their proof strategy for our proofs.

In Section 2.2.2, we prove PSN for a subcalculus of Λ_{sub} without certain interactions between substitutions. We identify the subset $\mathrm{SN}_{\Lambda_{\mathrm{sub}}}$ of strongly normalising terms of Λ_{sub} and show it to be a strict subset of the strongly normalising terms of $\lambda\mathrm{xgc}$. We then give a somewhat complicated, inductive proof of PSN.

Milner's calculus allows 'inter-substitution reduction', a reduction related to the notion of composition of substitutions which allows free occurences of variables in substitutions to be substituted for. This form of reduction is discussed throughout Section 2.2. It is explicit in the reduction rules which do not require a free occurrence of a variable to be located 'near' the substitution definition. This is possible in $'\Lambda_{\mathrm{BIG}}$ due to the bigraphical linking structure (with multiple locality) and wide reaction rules. The wide substitution rule is unusual compared to traditional explicit substitution calculi which typically have a set of distribution rules which serve to distribute substitution through a term to the variable level. In Section 2.3, we note that Λ_{sub} manages to mimic full composition of substitutions. In Section 3.1 we propose an extension to Λ_{sub} (based on $\lambda\mathrm{xc}^-$) which allows a direct weak composition of substitutions. We propose that the extension retains confluence and PSN but do not support that here.

The bigraphical encoding of Λ_{sub} shares similarities to the $\lambda\mathrm{lxr}$ calculus of Kesner and Lengrand [KL05, KL] which is worth investigating as $\lambda\mathrm{lxr}$ is the first explicit substitution calculi with confluence, preservation of strong normalisation (PSN), and full composition of substitutions. We review the similarities in Section 1.3. We then introduce a modified version of $\lambda\mathrm{lxr}$ which retains PSN. A translation from Λ_{sub} to this modified calculus is given which results in a neater proof of PSN.

This work does not directly concern bigraph theory. Milner introduced his bigraphical encoding of the λ-calculus both as an application of local bigraphs and as a starting point for studying confluence in bigraphical reactive systems (BRSs). Although we prove closed confluence (confluence on terms without metavariables) of $'\Lambda_{\mathrm{BIG}}$, this work does not add to a general theory of confluence for bigraph theory. However, the technical tools employed by Bloo and Rose in their proofs of closed confluence – projection and the generalised interpretation method – may be helpful in proving this property in other bigraphical explicit substitution encodings of confluent calculi.

This report is (probably overly) long. An interest-oriented reading order is shown in Figure 1.

Terminology and notation

We will assume that the reader is familiar with bigraph theory [JM04, Mil05b], the λ-calculus [Bar84], and rewrite systems [Ter03]. All the required background reading for this work can be found in [Mil05b, Mil04] and [Ros96a]. The main technical work (Sections 2.1, 2.2, and 2.4) can be read without much knowledge of bigraph theory.

The notation used here follows [Ros96a] and [Mil04]. For bigraphs, $\doteq$ denotes support-equivalence and $\circ$ denotes composition. Λ denotes the set of λ-calculus terms and we use $\equiv$ to denote α-equivalence of λ-terms.

The bulk of the notation concerns rewrite systems and rewrite rules. We call these concepts *reduction* systems and *reduction* rules respectively as does Rose [Ros96a] following Huet [Hue80]. We will mainly be working with

[1] At the time of writing, this work was being revised.

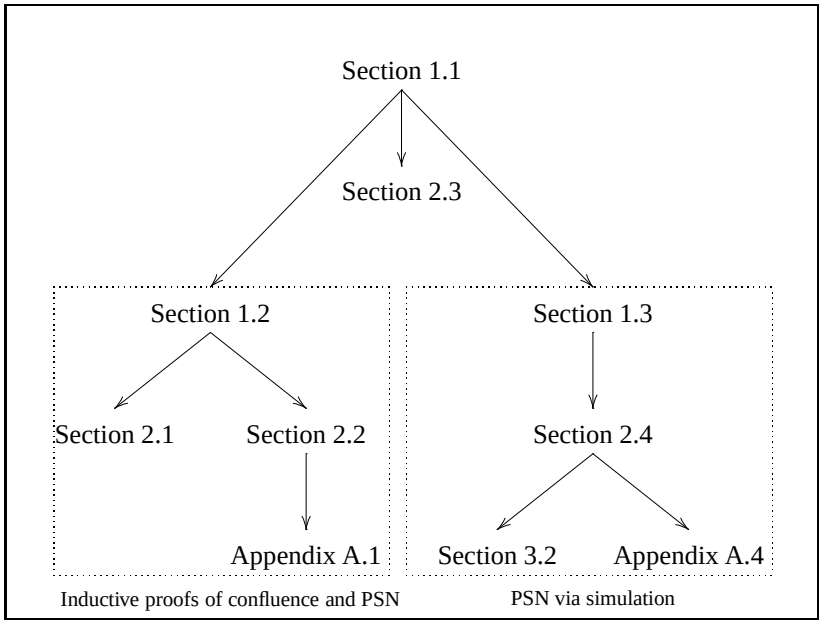
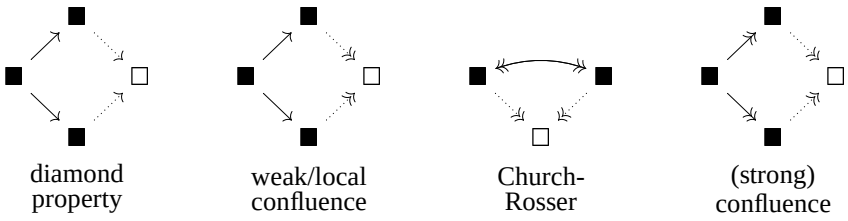


Figure 1: Interest-oriented reading order

variants of the λ-calculus and will use M, N, P, Q, and R to denote terms, u, v, w, x, y, and z to denote variables, and C and D to denote contexts. In both Λ_{sub} and $\lambda\mathrm{xgc}$, the set of terms (which is identical for both, assuming a variable convention introduced in the next section) is denoted by $\Lambda\mathrm{x}$.

When a reduction relation $\longrightarrow$ has a unique normal form, we write $\downarrow(M)$ for the $\longrightarrow$-normal form of term M. The symbol $\xrightarrow{n}$ denotes n consecutive reductions of $\longrightarrow$. The transitive closure of $\longrightarrow$ is denoted by $\xrightarrow{+}$, its reflexive closure is denoted by $\xrightarrow{=}$, and its transitive and reflexive closure is denoted by $\twoheadrightarrow$. When $\longrightarrow$ has unique normal forms, $\twoheadrightarrow\!\!\downarrow$ denotes 'reduction to normal form'. We write $\longrightarrow\mathrm{SN}$ when $\longrightarrow$ is strongly-normalising, $\longrightarrow\mathrm{CR}$ when it has the Church-Rosser property, $\longrightarrow\mathrm{LC}$ when it has weak confluence, $\longrightarrow\Diamond$ when it has the diamond property, and $\longrightarrow\mathrm{UN}$ when it has unique normal forms.

We only consider confluence on terms without metavariables – closed confluence – in this work and typically omit 'closed' from now on. The following diagrams depict the terminology we use when discussing confluence.



Weak confluence states that when a term a can reduce to two other terms b_1 and b_2 then there exists a common reduct, c, of the latter pair where $b_1 \twoheadrightarrow c$ and $b_2 \twoheadrightarrow c$. The other properties are defined similarly. Milner [Mil04] calls the diamond property *one-step confluence* and local confluence has also been called *weak Church-Rosser* or *weak confluence*. Any relation is confluent if and only if it has the Church-Rosser property (see [Ros96a] for a proof) and so we will use these terms interchangeably.

We will make one change from the presentation of Λ_{sub} in [Mil04], representing an explicit substitution with angle brackets as opposed to square brackets. We make the same change for the presentation of explicit substitutions in $\lambda\mathrm{lxr}$. This choice of notation was made both to match that used by Bloo and Rose for $\lambda\mathrm{xgc}$ and to avoid confusion with substitution in the pure λ-calculus.

Chapter 1

Introduction to the calculi

1.1 The calculus Λ_{sub} and its encoding

We will briefly introduce Λ_{sub} and then comment on its encoding, $'\Lambda\mathrm{BIG}$, into bigraphs. We refer the reader to [Mil04] for a full introduction to Λ_{sub} and $'\Lambda\mathrm{BIG}$. We do not discuss $'\Lambda\mathrm{BIG}$ much here. We will start by giving the inductive definition of the set $\Lambda\mathrm{x}$ of Λ_{sub} terms as:

$$M ::= x \quad | \quad \lambda x.M \quad | \quad MN \quad | \quad M\langle x := N\rangle$$

where the notation $\langle x := N\rangle$ represents an explicit substitution *i.e.* $M\langle x := N\rangle$ is a *term construction* and should not be confused with the substitution meta-operation of the λ-calculus. We will refer to this set of terms as $\Lambda\mathrm{x}$. It is also the set of terms of $\lambda\mathrm{xgc}$.

The definition of free variables of terms is given as part of the translation of Λ_{sub} into $'\Lambda\mathrm{BIG}$ and so we introduce that first.

Definition (λ-terms into bigraphs [Mil05b]).

$$\begin{aligned}
[\![x]\!]_{a,X\uplus x} &\stackrel{\text{def}}{=} \mathrm{var}_{ax} \oplus X \\
[\![\lambda x.M]\!]_{a,X} &\stackrel{\text{def}}{=} (\mathrm{lam}_{a(bx)} \oplus \mathrm{id}_X)[\![M]\!]_{b,X\uplus x} \\
[\![MN]\!]_{a,X} &\stackrel{\text{def}}{=} (\mathrm{app}_{a(bc)} \oplus \mathrm{id}_X)([\![M]\!]_{b,X} \mid [\![N]\!]_{c,X}) \\
[\![M\langle x := N\rangle]\!]_{a,X} &\stackrel{\text{def}}{=} \big(\mathrm{sub}_{a(bdx)} \oplus \mathrm{id}_X\big)\big([\![M]\!]_{b,X\uplus x} \mid (\mathrm{def}_{dx(c)} \oplus \mathrm{id}_X)[\![N]\!]_{c,X}\big).
\end{aligned}$$

The second index of a translation $[\![M]\!]_{a,X}$ is a set which must at least contain the free variables of M in Λ_{sub}. X may contain other names. This is necessary for technical reasons; in a bigraphical reactive system, the reactum of a rule must have the same set of free names as the redex. Therefore if a rule discards some free variables, the names of those variables persist through the reduction. For example, in the reduction

$$[\![x\langle y := z\rangle]\!]_{a,\{x,z\}} \xrightarrow{\mathrm{D}} [\![x]\!]_{a,\{x,z\}}$$

with all variables distinct, the reduct retains the free name z.

The translation allows us to derive the set of free variables of a $\Lambda\mathrm{x}$ term in Λ_{sub} as:

$$\begin{aligned}
\mathrm{fv}(x) &= \{x\} \\
\mathrm{fv}(\lambda x.M) &= \mathrm{fv}(M) \setminus \{x\} \\
\mathrm{fv}(MN) &= \mathrm{fv}(M) \cup \mathrm{fv}(N) \\
\mathrm{fv}(M\langle x := N\rangle) &= (\mathrm{fv}(M) \cup \mathrm{fv}(N)) \setminus \{x\}
\end{aligned}$$

The last definition may be misleading. The translation from Λ_{sub} to $'\Lambda\mathrm{BIG}$ highlights an implicit variable convention in Λ_{sub}. If we examine the translation, we see that the set of free variables of N is disjoint from $\{x\}$. This last definition can therefore be written

$$\mathrm{fv}(M\langle x := N\rangle) = (\mathrm{fv}(M) \setminus \{x\}) \cup \mathrm{fv}(N)$$

as $x \notin \mathrm{fv}(N)$. This convention is important to avoid variable capture (see below).

The reduction rules of Λ_{sub}, which we name after their encodings in $'\Lambda\mathrm{BIG}$, are presented below in the manner of [Ros96a].

Definition (Reduction rules for Λ_{sub}). *Define the following reductions on the set of Λ_{sub} terms (more precisely, their contextual closure modulo $\equiv$).*

1. $\xrightarrow[\mathrm{A}]{}$, *Apply* (substitution generation)
$$(\lambda x.M)N \to M\langle x := N\rangle \qquad \text{(A)}$$

2. $\xrightarrow[\mathrm{C}]{}$, *Copy* (wide/non-local/distant/external (explicit) substitution)
$$M\langle x := N\rangle \to M'\langle x := N\rangle \quad \text{if } x \in \mathrm{fv}(M) \qquad \text{(C)}$$
where $M = C[x]$ and $M' = C[N]$ for some Λx-context C where this occurrence of x is free.

3. $\xrightarrow[\mathrm{D}]{}$, *Discard* (garbage collection)
$$M\langle x := N\rangle \to M \quad \text{if } x \notin \mathrm{fv}(M) \qquad \text{(D)}$$
As in [Ros96a], the subterm N in $M\langle x := N\rangle$ is called *garbage* if $x \notin \mathrm{fv}(M)$. This rule also exists in λxgc and is called *explicit* garbage collection.

4. Λ_{sub}-reduction is $\xrightarrow[\mathrm{ACD}]{} = \xrightarrow[\mathrm{A}]{} \cup \xrightarrow[\mathrm{C}]{} \cup \xrightarrow[\mathrm{D}]{}$. We will also study the relation $\xrightarrow[\mathrm{CD}]{} = \xrightarrow[\mathrm{C}]{} \cup \xrightarrow[\mathrm{D}]{}$ and other combinations in later sections.

We choose to call $\xrightarrow[\mathrm{C}]{}$ *wide substitution* as the corresponding parametric reaction rule of $'\Lambda\mathrm{BIG}$ is wide – it takes a bigraph with width two as its parameter. Milner describes such rules as *non-local* [Mil05a], and $\xrightarrow[\mathrm{C}]{}$ as substitution acting 'at a distance' [Mil05b]. In Λ_{sub}, the upshot is that in a $\xrightarrow[\mathrm{C}]{}$ redex, the free occurrence of the variable may be located apart from the substitution definition. This is in contrast to the distributive, *local* rules of most traditional explicit substitution calculi. Non-local reactions allow full composition of substitutions in Λ_{sub}, a feature which we will discuss particularly in the proof of PSN. Klop has called similar non-local substitutions in term graph rewrite systems *external substitutions* [Klo95].

In Λ_{sub}, a single β-reduction is imitated with $n + 2$ reductions (where n is the number of free occurrences of the variable to be substituted for) so that

$$\text{if } M \xrightarrow[\beta]{} N \text{ then } M \xrightarrow[\mathrm{A}]{} \xrightarrow[\mathrm{C}]{}^{n} \xrightarrow[\mathrm{D}]{} N.$$

As with most explicit substitution calculi, the reductions which combine to imitate β-reduction may overlap (see Appendix A.3 for an example). This behaviour can lead to calculi where confluence or PSN is not guaranteed. Melliès' counterexample to PSN for $\lambda\sigma$ [Mel95] was unexpected and led to the study of other calculi which satisfy PSN. The work of Bloo, Rose, and Geuvers [BR95, Ros96a, Ros96b, Blo97, BG99] was instrumental in this line of research and is the basis for the proofs in Sections 2.1 and 2.2.

The rest of this section will discuss how bigraph theory can capture some of the important notions of the λ-calculus and how $'\Lambda\mathrm{BIG}$ relates to previous work. Some of the following also applies to the encoding of other calculi in bigraphs.

Variable convention In the β reduction below, the free occurence of x in $(\lambda x.\lambda y.x)$ is replaced by the argument y. To avoid variable capture, one typically employs a variable convention where the bound variables of a term are chosen to be different and distinct from the free variables of the term.

$$(\lambda x.\lambda y.x)y \xrightarrow[\beta]{} \equiv_\alpha \lambda z.y$$

If $(\lambda x.\lambda y.x)y$ was a valid Λ_{sub} term then we would have the reduction sequence:

$$(\lambda x.\lambda y.x)y \xrightarrow[\mathrm{A}]{} (\lambda y.x)\langle x := y\rangle \xrightarrow[\mathrm{C}]{} (\lambda y.y)\langle x := y\rangle \xrightarrow[\mathrm{D}]{} (\lambda y.y).$$

However, if we follow the translation rules, we get:

$$\begin{aligned}
& [\![(\lambda x.\lambda y.x)y]\!]_{a,y\uplus Y} \\
= \;& (\mathrm{app}_{a(bc)} \oplus \mathrm{id}_X)([\![\lambda x.\lambda y.x]\!]_{b,y\uplus Y} \mid (\mathrm{var}_{cy} \oplus Y)) \\
& \text{where} \\
& [\![\lambda x.\lambda y.x]\!]_{b,y\uplus Y} \\
= \;& (\mathrm{lam}_{b(dx)} \oplus \mathrm{id}_{y\uplus Y})[\![\lambda y.x]\!]_{d,y\uplus Y\uplus x} \\
= \;& (\mathrm{lam}_{b(dx)} \oplus \mathrm{id}_{y\uplus Y})(\mathrm{lam}_{d(ey)} \oplus \mathrm{id}_{y\uplus Y\uplus x})[\![x]\!]_{f,y\uplus Y\uplus x\uplus y} \\
= \;& \big(\mathrm{lam}_{b(dx)} \oplus \mathrm{id}_{y\uplus Y}\big)\big(\mathrm{lam}_{d(ey)} \oplus \mathrm{id}_{y\uplus Y\uplus x}\big)\big(\mathrm{var}_{fx} \oplus (y \uplus Y \uplus y)\big).
\end{aligned}$$

Since y can not be disjoint from itself, we therefore assume that in Λ_{sub}, the bound names of a term are distinct from the free names.

This variable convention implicit in the translation from Λ_{sub} to $'\Lambda$BIG also means that the set of free variables of a term is preserved by $\xrightarrow[\mathrm{A}]{}$ reduction. Consider the following translation.

$$\begin{aligned}
& [\![(\lambda x.M)N]\!]_{a,X} \\
= \;& (\mathrm{app}_{a(bc)} \oplus \mathrm{id}_X) \circ ([\![(\lambda x.M)]\!]_{b,X} \mid [\![N]\!]_{c,X}) \\
= \;& \big(\mathrm{app}_{a(bc)} \oplus \mathrm{id}_X\big) \circ \big((\mathrm{lam}_{b(dx)} \oplus \mathrm{id}_X)[\![M]\!]_{d,X\uplus x} \mid [\![N]\!]_{c,X}\big)
\end{aligned}$$

As x is disjoint from X, it is not a free variable of N. This term reduces to

$$\begin{aligned}
\xrightarrow[\mathrm{A}]{} \;& \big(\mathrm{sub}_{a(dbx)} \oplus \mathrm{id}_X\big) \circ \big([\![M]\!]_{d,X\uplus x} \mid (\mathrm{def}_{bx(c)} \oplus \mathrm{id}_X)[\![N]\!]_{c,X}\big) \\
= \;& [\![M\langle x := N\rangle]\!]_{a,X}
\end{aligned}$$

where the entire term, M, and N respectively have the same set of free variables as in $(\lambda x.M)N$.

Similarly, one can show that $(\lambda x.\lambda x.M)$ is not a valid term in Λ_{sub} and we can therefore assume that nested abstractions or substitutions must have different bound names. We could take the stance that bound variables in a term do not have to be distinct as the terms $(\lambda x.z)(\lambda x.z)$ and $M\langle x := \lambda x.M\rangle$ both have valid translations. However, as α-equivalent terms have equivalent translations and reaction in $'\Lambda$BIG matches reduction in Λ_{sub} [Mil04], we assume the following convention, following Barendregt [Bar84].

Convention (bound variables in Λ_{sub}). *When working in Λ_{sub}, we assume that the bound names of a term are distinct and different from the free names of the term.*

Bound variables In a $'\Lambda$BIG term, a variable x is bound by linking a λ-port from a var- or def-node to a (binding) λ-port of a lam- or sub-node. This binding ensures that the variable cannot be accessed outside of the node on which the binding port lies *i.e.* we have a sort of name-scoping. This feature of bigraphs fits neatly with the λ-calculus.

Further, this bound link is an *edge*. An edge in a bigraph forms part of the support set of the bigraph. Two bigraphs are said to be support-equivalent when they are the same up to an isomorphism of their support sets. Hence, support-equivalence naturally encodes α-equivalence for the λ-calculus.

Instantiation Replication of terms in the λ-calculus *e.g.* $(\lambda x.xx)N \xrightarrow{}_{\beta} NN$ and destruction of terms *e.g.* $((\lambda x.\lambda y.y)M)N \xrightarrow{}_{\beta}{}^{*} N$ can be encoded in bigraphs via the notion of instantiation. Roughly speaking, instantiation is the bigraphical mechanism which allows the parameters of the redex in a parametric reaction rule to be discarded or copied in the reactum. It is also crucial for encoding the π-calulus as it allows replication.

Abstract syntax tree A bigraph may be partially represented by a sort of abstract syntax tree based on the place graph, where composition represents nesting and prime product represents branching. For example, the encoding of a λ-term in $'\Lambda$BIG can be represented as in Figure 1.1. The dotted lines link free occurrences to names (at the top of the figure) or represent a binding (the curved line).

With this representation, the abstract syntax tree of a λ-term closely matches this 'abstract syntax tree of composition' in $'\Lambda$BIG, suggesting that the encoding is indeed very natural and not forced.

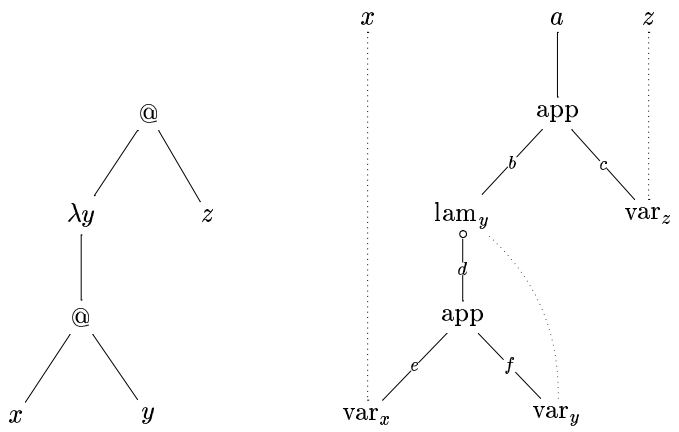


Figure 1.1: Abstract syntax trees for $(\lambda y.xy)z$ and a representation of the corresponding bigraph $\mathrm{app}_{a(bc)} \circ (\mathrm{lam}_{b(dy)} \circ \mathrm{app}_{d(ef)} \circ (\mathrm{var}_{ex} \mid \mathrm{var}_{fy}) \mid \mathrm{var}_{cz})$

Explicit substitution Explicit substitution in $'\Lambda_{\mathrm{BIG}}$ is represented with the help of a pair of controls, def and sub. A β-reduction $(\lambda x.M)N \to M[x := N]$ is encoded as a three-part reduction sequence. First, an explicit substitution is generated, introducing a def node and a sub node into the bigraph. The def node encompasses the term N, representing 'this is to be copied.' The sub node both takes a hold of all free occurrences of x in M and encompasses both M and the def node. Next, all the free variables x in M are substituted with N, one at a time. Informally, the sub node picks one of its free xs, throws it away and puts the contents of the def node (N) in its place. Finally, the explicit substitution is discarded (garbage-collected) *i.e.* the sub node is thrown away and the def node and its contents are thrown away. We are left with a bigraph representing $M[x := N]$. This description is simplistic as these non-atomic (in the sense of β-reduction) reductions may interleave (see Appendix A.3).

The term $M\langle x := N\rangle$ can also be described as 'M placed in a substitution context where x is defined to be N'. This description is represented in the encoding by placing $[\![M]\!]$ inside a sub node where all free xs in M are connected to a def node containing $[\![N]\!]$.

The combination of names with multiple locality, wide reactive rules, and active controls means that a free occurrence of x anywhere below a substitution $\langle x := N\rangle$ may be replaced by N. Specifically, this allows substitutions to replace such occurrences in substitutions below them *e.g.*

$$N\langle x := y\rangle\langle y := P\rangle \xrightarrow[\mathrm{C}]{} N\langle x := P\rangle\langle y := P\rangle.$$

This behaviour is known as *composing substitutions*, is not present in all explicit substitution calculi, and may break PSN. For example, this behaviour breaks PSN for $\lambda\sigma$ and is not allowed in $\lambda\mathrm{xgc}$ (which has PSN). Calculi exist which have both PSN and composition of substitutions in some form but few have both PSN and *full composition of substitutions* as Λ_{sub} has (see Section 2.3). We will show that Λ_{sub} and $'\Lambda_{\mathrm{BIG}}$ both have full composition of substitutions, can simulate β-reduction, are confluence, and have PSN. To date and to our knowledge, the only other explicit substitution calculus which has these properties is $\lambda\mathrm{lxr}$, introduced later.

Relating $'\Lambda_{\mathrm{BIG}}$ to previous work in the π-calculus There are – perhaps unsurprisingly – some nice similarities between the encoding of the lazy λ-calculus in the π-calculus [Mil90] and $'\Lambda_{\mathrm{BIG}}$.

In the former, abstractions are encoded as:

$$[\![\lambda x.M]\!]u \stackrel{\mathrm{def}}{=} u(x).u(v).[\![M]\!]v$$

where the names x and v are bound. In this encoding, the names u and v are used as a means of tagging terms and passing them around.

In $'\Lambda_{\mathrm{BIG}}$, abstractions are encoded as:

$$[\![\lambda x.M]\!]_{u,X} \stackrel{\mathrm{def}}{=} (\mathrm{lam}_{u(vx)} \oplus \mathrm{id}_X)[\![M]\!]_{v,X \uplus x}$$

where again x and v are bound. In this encoding, the names u and v are called tag-names and x is called a λ-name.

For the encoding of variables; in the π-calculus, $[\![x]\!]u \stackrel{\text{def}}{=} \bar{x}u$ while in $'\Lambda$BIG the translation is $\mathrm{var}_{ux} \oplus X$ for some set X. In both cases, the names u and x are free.

The encoding of an 'environment entry' used for substitution in the π-calculus is:

$$[\![x := M]\!] \stackrel{\text{def}}{=} !x(w).[\![M]\!]w$$

This replicated term keeps the substitution definition alive, within the scope of the substitution, until no more substitutions may be performed. Then, the replicated term may be garbage-collected through strong bisimilarity. This is similar to the case in $'\Lambda$BIG where instantiation keeps the explicit substitution alive by copying M and garbage-collection is represented with an explicit reduction rule.

An interesting point is that variables of the λ-calculus are treated as controls in $'\Lambda$BIG whereas in the bigraphical encoding of the asynchronous π-calculus [JM04], names are treated as names in the encoding. This would appear quite sensible as a variable of the λ-calculus is also a λ-term while the same is not true of a π-calculus name and the two calculi are built around different paradigms. However, we feel it is helpful to collect intuitions on these encoding choices.[1]

Reduction strategies Another interesting feature of $'\Lambda$BIG, arising from the general bigraph theory, is that reductions may be applied to any subterm of a term. This includes, as with λxgc, substitution inside an explicit substitution *i.e.*

$$\text{if } N \longrightarrow N' \text{ then } M\langle x := N\rangle \longrightarrow M\langle x := N'\rangle.$$

Aside from the full evaluation strategy [Bar84] of the λ-calculus (Figure 1.2), other strategies include the lazy

$$\frac{}{(\lambda x.M)N \longrightarrow M[x := N]}\ (\beta) \qquad \frac{M \longrightarrow M'}{MN \longrightarrow M'N}\ (\nu)$$

$$\frac{M \longrightarrow M'}{(\lambda x.M) \longrightarrow (\lambda x.M)}\ (\xi) \qquad \frac{N \longrightarrow N'}{MN \longrightarrow MN'}\ (\mu)$$

Figure 1.2: Full evaluation strategy for the λ-calculus

reduction strategy, having the rules β and ν, and the strong lazy reduction strategy, having the rules β, ν and ξ.

Milner [Mil90] showed that the π-calculus could encode the lazy λ-calculus and the call-by-value λ calculus. Parrow and Victor [PV98] later showed that the fusion calculus could encode the strong lazy λ-calculus through a slight modification of the encoding in [Mil90], utilising the symmetry in their calculus.

Since bigraphs build on ideas from both of these calculi, we would like it to be able to encode at least the strong lazy λ-calculus. It turns out that Λ_{sub}, and hence $'\Lambda$BIG, encodes the full evaluation strategy. This is shown in Theorem 8, Section 2.1, which states: for λ-terms M and N,

$$M \xrightarrow[\mathrm{ACD}]{}\!\!\!\!\!\gg N \iff M \xrightarrow[\beta]{}\!\!\!\!\!\gg N$$

so that β-reduction may be simulated step-by-step in $'\Lambda$BIG.

[1] This idea of 'design patterns for bigraphs' is something that Troels C. Damgaard has discussed on several occasions.

1.2 The calculus $\lambda\mathrm{xgc}$

$\lambda\mathrm{xgc}$ [BR95, Ros96b, Blo97], due to Bloo and Rose, is a refinement of the λ-calculus in the tradition of $\lambda\sigma$ [ACCL91]. It is an explicit substitution calculus with (explicit) garbage collection.

The motivation behind comparing Λ_{sub} and $\lambda\mathrm{xgc}$ is as follows. Λ_{sub} uses explicit substitution, names as opposed to de Bruijn notation in its inference rules, and a form of *explicit* garbage collection [Ros92] so it seems sensible to compare Λ_{sub} to a version of the λ-calculus which has these properties, as $\lambda\mathrm{xgc}$ has.

$\lambda\sigma$ [ACCL91] has the first of these properties but lacks the notion of explicit garbage collection inherent in $\lambda\mathrm{xgc}$ [Ros96a]. A variant of $\lambda\sigma$ using names was discussed in section 3.3 of [ACCL91] but has some complications as described in that work and in [Ros96a]. As a result, [ACCL91] concentrates mostly on a presentation using de Bruijn notation.

There are some immediate similarities between Λ_{sub} and $\lambda\mathrm{xgc}$. They both have the same set of terms (assuming variable conventions). The definition of free variables is identical. In terms of reaction/reduction relations, $\xrightarrow[\mathrm{A}]{}$ and $\xrightarrow[\mathrm{D}]{}$ respectively match $\xrightarrow[\mathrm{b}]{}$ and $\xrightarrow[\mathrm{gc}]{}$ from [Ros96a] exactly. Further, in $\lambda\mathrm{xgc}$, reductions are allowed inside explicit substitutions.

One crucial difference between the calculi is that $\lambda\mathrm{xgc}$ propagates explicit substitutions to the variable level of a term (see the definition below), inducting over the structure, whereas substitution can act at a distance in Λ_{sub}. Λ_{sub} also allows full composition of substitutions whereas $\lambda\mathrm{xgc}$ does not allow substitutions to be composed (although extensions of $\lambda\mathrm{xgc}$ do – see Sections 2.3.3-2.3.5). We will see later that this composition in Λ_{sub} means that its set of strongly normalising terms is a subset of that of $\lambda\mathrm{xgc}$.

Both calculi share a rule for garbage collection at a distance – explicit garbage collection. $\lambda\mathrm{xgc}$ has a local garbage-collection rule $\xrightarrow[\mathrm{xvgc}]{}$ which begs the question why a non-local rule is useful. One answer is that $\xrightarrow[\mathrm{gc}]{}$ allows for short-cuts where a useless substitution (garbage) does not propagate through a term but is discarded immediately. However, Rose [Ros96a] states that the main technical reason for the existence of explicit garbage collection in $\lambda\mathrm{xgc}$ is that it greatly eases the proof of PSN. It seemed reasonable to be able to adapt their work to prove these properties for Λ_{sub}, especially since the proof of PSN rested upon $\xrightarrow[\mathrm{gc}]{}$.

The reduction rules of $\lambda\mathrm{xgc}$ are as follows.

Definition ($\lambda\mathrm{xgc}$ reduction). *$\xrightarrow[\mathrm{x}]{}$, explicit substitution, is the contextual closure (modulo $\equiv$) of the union of*

$$x\langle x := N\rangle \to N \tag{xv}$$

$$x\langle y := N\rangle \to x, \quad \textit{if } x \neq y \tag{xvgc}$$

$$(\lambda x.M)\langle y := N\rangle \to \lambda x.M\langle y := N\rangle \tag{xab}$$

$$(M_1 M_2)\langle y := N\rangle \to M_1\langle y := N\rangle M_2\langle y := N\rangle \tag{xap}$$

$\lambda\mathrm{xgc}$-reduction is $\xrightarrow[\mathrm{bxgc}]{} = \xrightarrow[\mathrm{b}]{} \cup \xrightarrow[\mathrm{x}]{} \cup \xrightarrow[\mathrm{gc}]{}$ where $\xrightarrow[\mathrm{b}]{} = \xrightarrow[\mathrm{A}]{}$ and $\xrightarrow[\mathrm{gc}]{} = \xrightarrow[\mathrm{D}]{}$.

The condition that $x \neq y$ is implicit in the (xab) rule as the variable convention is also employed for $\lambda\mathrm{xgc}$.
The main differences between $\xrightarrow[\mathrm{ACD}]{}$ and $\xrightarrow[\mathrm{bxgc}]{}$ are that:

(i) $\xrightarrow[\mathrm{C}]{}$ enables substitutions to be performed without migrating the explicit substitutions inside terms as $\lambda\mathrm{xgc}$ does via the xab and xap rules. This wide substitution is made possible partly due to the linking structure of bigraphs which connects a substitution to free occurences of the variable to be replaced. This is certainly a nice feature of bigraphs but it also prevents us from using some inductive techniques that Bloo and Rose employed in their proofs. As a result, some of our proofs in Sections 2.1 and 2.2 are necessarily different or more verbose from the originals.

(ii) $\xrightarrow[\mathrm{C}]{}$ allows inter-substitution reduction (composition of substitutions) *e.g.*

$$N\langle x := y\rangle\langle y := P\rangle \xrightarrow[\mathrm{C}]{} N\langle x := P\rangle\langle y := P\rangle.$$

This form of reduction is not possible in $\lambda\mathrm{xgc}$ which can only push substitutions inside applications or abstractions. This has consequences for the proof of PSN later but for now we will just remark that the set of strongly normalising terms of $\xrightarrow[\mathrm{ACD}]{}$ is smaller than that of $\xrightarrow[\mathrm{bxgc}]{}$.

(iii) Given a term $x\langle x := N\rangle$, $\lambda\mathrm{xgc}$ may perform a single xv step to reach the term N whereas Λ_{sub} must perform a C step followed by a D step – $\xrightarrow[\mathrm{xv}]{}$ automatically garbage collects having replaced the only free

occurrence of a variable whereas C does not garbage collect as in the general case there may be more free occurrences of x under the substitution.

(iv) The xvgc rule becomes redundant in Λ_{sub} since if $M \xrightarrow{\mathrm{xvgc}} N$ in $\lambda\mathrm{xgc}$ then $M \xrightarrow{\mathrm{gc}} N$ in Λ_{sub} (and $\lambda\mathrm{xgc}$).

For an intuition into comparing substitutions in the calculi, note that given a term M such that $M \xrightarrow{\mathrm{x}}\!\!\!\!\!\rightarrow N$ we have that $M \xrightarrow{\mathrm{CD}}\!\!\!\!\!\rightarrow N$ also *i.e.* the normal forms of $\xrightarrow{\mathrm{x}}$ and $\xrightarrow{\mathrm{CD}}$ coincide.

Finally, Martin Elsman shared this intuition with me:

"Informally, one can think of reductions in $\lambda\mathrm{xgc}$ involving explicit substitutions to encode some kind of 'reference counting' whereas in Λ_{sub}, garbage collection is closer to 'reference tracing' as reductions involve finding all free variables $\mathrm{fv}(N)$ of a term N in $N\langle x := M\rangle$.

"On the other hand, substitution distribution in $\lambda\mathrm{xgc}$ also involves traversing the term, although this is done lazily!"

1.3 The calculus λlxr

λlxr [KL05, KL] is an explicit substitution calculus which is a sound and complete computational counterpart to the intuitionistic part of the Proof Nets of Linear Logic [Gir87]. It is also the first published explicit substitution calculus to enjoy the properties of confluence, PSN, and full composition of substitutions. We concentrate on these properties here and do not review the typing system of λlxr or its connection to linear logic.

λlxr builds partly on work by David and Guillaume on the λ_{ws} calculus [DG01]. The λ_{ws} calculus allowed a level of composition of substitutions whilst retaining PSN and was one of the first explicit substitution calculi which satisfied step-by-step simulation of β-reduction, confluence on terms with metavariables, and PSN.

The set of terms of λlxr is similar to that of λxgc. It is defined (with a slight change of notation) by:

$$M ::= x \mid \lambda x.M \mid MN \mid M\langle x := N\rangle \mid W_x(M) \mid C_x^{y,z}(M)$$

The two new constructors are $W_x(M)$, an *explicit weakening*, and $C_x^{y,z}$, an *explicit contraction*. The sets of free variables are as before for the old constructors. x is free in $W_x(M)$ and $C_x^{y,z}$ whereas y and z are bound in the latter. Again, we follow a variable convention where each bound name of a term M is distinct and different from any free names in M.

We now discuss three important features in λlxr – weakenings, contractions, and linearity of terms[2].

The term $W_x(M)$ is an annotated form of M which states that the free variable x does not occur free in M. As it is explicitly part of the syntax, it can play a rôle in the reduction relation of λlxr and weakenings are in fact used to provide an explicit garbage collection rule. For example, consider the term $W_x(M)\langle x := N\rangle$. As x does not occur free in M, we may want to garbage collect the substitution. The rule $(Weak1)$ in Figure 1.4 does precisely this. Kesner and Lengrand note that weakenings in λlxr may always be pulled out to the top level allowing efficient garbage collection whereas λ_{ws} cannot pull its labels out to the top-level. We note that in $'\Lambda$BIG, substitutions are never propagated through terms and garbage collection is always at its most efficient.

Substitution in λlxr is defined as a set of distributive rules. Weakenings also allow efficient propagation of substitutions. For example, propagating the substitution $x := N$ through $W_x(M)$ is pointless as no substitution can take place. Indeed, the reduction rules do not permit this propagation.

Another interesting property of weakenings is that they allow free variables to be kept through reduction. The two destructive rules are (Var) and $(Weak1)$. As expected, the substitution rule (Var) does not lose free variables. Interestingly, the garbage collection rule $(Weak1)$ remembers the free variables of the discarded substitution via a weakening. Kesner and Lengrand compare this preservation of free variables to "interface preserving"[Laf90] in interaction nets[3]. This property is also present in $'\Lambda$BIG where, due to the preservation of outer interfaces through reduction, free variables are remembered through reaction.

Contractions in λlxr allow the linearity of terms discussed below. The term $C_x^{y,z}(M)$ may be read as 'M where y and z are x.'

Terms in λlxr may always be assumed to be linear. A term M is linear if "*in every subterm, every variable has at most one free occurrence, and every binder binds a variable that does have a free occurrence (and hence only one)*" [KL05]. It is possible to translate every λ-term to a (linear) λlxr term. This linearity appears to be a large factor in allowing λlxr to retain PSN whilst having full composition of substitutions (FCS). The $\overrightarrow{\text{xap}}$ rule in λxgc duplicates substitutions unconditionally and possibly unnecessarily – free occurrences of the bound variable may not exist in either branch of the application. This unconditional copying of substitutions features in the counterexamples of PSN in Section 2.3. The $(Cont1)$ rule which copies substitutions in λlxr does so conditionally and out of necessity – there is always guaranteed to be exactly two free occurrences of y and z below the contraction $C_x^{y,z}$.

The congruence axioms and reduction rules for λlxr can be found in Figures 1.3 and 1.4 respectively. The congruence axioms were chosen to strengthen the relationship between λlxr and Proof Nets and we consider them in the next section in terms of equivalence in $'\Lambda$BIG.

In the reduction rules, the notation $R_\Delta^\Phi(M)$, where Φ and Δ are finite lists (with no repetition) of distinct variables and equal length, denotes the result of simultaneously replacing $x \in \Phi$ in M with $y \in \Delta$ where both variables occur as the i^{th} variable in their respective lists. The meta-notation $W_{\text{fv}(N)}$ and $C_\Phi^{\Delta,\Pi}$ denotes multiple weakenings and contractions – the order is irrelevant up to congruences $\equiv_{C2_c}$ and $\equiv_{C_w}$.

Many of the reduction rules of λlxr (especially in System r) deal with pulling weakenings outwards and pushing contractions inwards. Linearity of terms means that substitutions are not replicated during propagation

[2] Fernández and Mackie [FM99] used these notions in earlier work.

[3] Bigraph theory has used ideas from interaction nets. We refer the reader to Leifer and Milner's arithmetic nets [LM04] and the bigraphical nets of Fernández, Mackie, and Sinot [FMS06] for further links between the theories.

$C_w^{x,v}(C_x^{z,y}(M))$	$\equiv_A$	$C_w^{x,y}(C_x^{z,v}(M))$	if $x \neq y, v$
$C_x^{y,z}(M)$	$\equiv_{C1_c}$	$C_x^{z,y}(M)$	
$C_{x'}^{y',z'}(C_x^{y,z}(M))$	$\equiv_{C2_c}$	$C_x^{y,z}(C_{x'}^{y',z'}(M))$	if $x \neq y', z' \& x' \neq y, z$
$W_x(W_y(M))$	$\equiv_{C_w}$	$W_y(W_x(M))$	
$M\langle x := P\rangle\langle y := Q\rangle$	$\equiv_S$	$M\langle y := Q\rangle\langle x := P\rangle$	if $y \notin \mathrm{fv}(P) \& x \notin \mathrm{fv}(Q)$ $\& \; x \neq y$
$C_w^{y,z}(M)\langle x := P\rangle$	$\equiv_{Cont2}$	$C_w^{y,z}(M\langle x := P\rangle)$	if $x \neq w \& y, z \notin \mathrm{fv}(P)$

Figure 1.3: Congruences for λlxr

$(\lambda x.M)N$	$\longrightarrow_B$	$M\langle x := N\rangle$	
System x			
$(\lambda y.M)\langle x := N\rangle$	$\longrightarrow_{Abs}$	$\lambda y.M\langle x := N\rangle$	
$(MN)\langle x := P\rangle$	$\longrightarrow_{App1}$	$M\langle x := P\rangle N$	$x \in \mathrm{fv}(M)$
$(MN)\langle x := P\rangle$	$\longrightarrow_{App2}$	$MN\langle x := P\rangle$	$x \in \mathrm{fv}(N)$
$x\langle x := M\rangle$	$\longrightarrow_{Var}$	M	
$W_x(M)\langle x := N\rangle$	$\longrightarrow_{Weak1}$	$W_{\mathrm{fv}(N)}(M)$	
$W_y(M)\langle x := N\rangle$	$\longrightarrow_{Weak2}$	$W_y(M\langle x := N\rangle)$	$x \neq y$
$M\langle x := P\rangle\langle y := Q\rangle$	$\longrightarrow_{Comp}$	$M\langle x := P\langle y := Q\rangle\rangle$	$y \in \mathrm{fv}(P)$
$C_x^{y,z}(M)\langle x := N\rangle$	$\longrightarrow_{Cont1}$	$C_\Phi^{\Delta,\Pi}(M\langle y := N_1\rangle\langle z := N_2\rangle)$	where $\Phi := \mathrm{fv}(N)$ $N_1 = R_\Delta^\Phi(N)$ $N_2 = R_\Pi^\Phi(N)$
System r			
$\lambda x.W_y(M)$	$\longrightarrow_{W\ Abs}$	$W_y(\lambda x.M)$	$x \neq y$
$W_y(M)N$	$\longrightarrow_{W\ App1}$	$W_y(MN)$	
$MW_y(N)$	$\longrightarrow_{W\ App2}$	$W_y(MN)$	
$M\langle x := W_y(N)\rangle$	$\longrightarrow_{W\ Subs}$	$W_y(M\langle x := N\rangle)$	
$C_w^{y,z}(W_y(M))$	$\longrightarrow_{Merge}$	$R_w^z(M)$	
$C_w^{y,z}(W_x(M))$	$\longrightarrow_{Cross}$	$W_x(C_w^{y,z}(M))$	$x \neq y, z$
$C_w^{y,z}(\lambda x.M)$	$\longrightarrow_{C\ Abs}$	$\lambda x.C_w^{y,z}(M)$	
$C_w^{y,z}(MN)$	$\longrightarrow_{C\ App1}$	$C_w^{y,z}(M)N$	$y, z \in \mathrm{fv}(M)$
$C_w^{y,z}(MN)$	$\longrightarrow_{C\ App2}$	$MC_w^{y,z}(N)$	$y, z \in \mathrm{fv}(N)$
$C_w^{y,z}(M\langle x := N\rangle)$	$\longrightarrow_{C\ Subs}$	$M\langle x := C_w^{y,z}(N)\rangle$	$y, z \in \mathrm{fv}(N)$

Figure 1.4: Reduction rules for λlxr

through a term unless a contraction is reached in which case the substitution is duplicated and the copies renamed to maintain linearity ($Cont1$). Besides these rules, the main ones corresponding roughly to $\overrightarrow{\text{ACD}}$ are substitution introduction (B), copying (Var), and explicit garbage collection ($Weak1$). There is then one reduction rule for explicit composition of substitutions ($Comp$). Note that this rule only takes care of the case $y \in \text{fv}(P)$ however, the other case $y \in \text{fv}(M)$ is taken care of by the $\equiv_S$ congruence (assuming linearity and the variable convention). This allows λlxr FCS.

The reduction relation $\longrightarrow_{\lambda\text{lxr}}$ of λlxr is the union of the reduction rules in Figure 1.4 modulo the congruences in Figure 1.3. To tie in with our remarks in Section 2.2.2 on intersubstitution reduction in Λ_{sub}, the λlxr reduction sequence below corresponds to the Λ_{sub} sequence $Z \overrightarrow{\text{ACD}}^{\gg} z\langle x := \Omega\rangle$.

$$
\begin{array}{rl}
 & z\langle x := uv\rangle\langle u := \lambda a.C_a^{b,c}(bc)\rangle\langle v := \lambda d.C_d^{e,f}(ef)\rangle \\
\longrightarrow_{(Comp)} & z\langle x := (uv)\langle u := \lambda a.C_a^{b,c}(bc)\rangle\rangle\langle v := \lambda d.C_d^{e,f}(ef)\rangle \\
\longrightarrow_{(Comp)} & z\langle x := (uv)\langle u := \lambda a.C_a^{b,c}(bc)\rangle\langle v := \lambda d.C_d^{e,f}(ef)\rangle\rangle \\
\longrightarrow_{(App1)} & z\langle x := (u\langle u := \lambda a.C_a^{b,c}(bc)\rangle v)\langle v := \lambda d.C_d^{e,f}(ef)\rangle\rangle \\
\longrightarrow_{(App2)} & z\langle x := u\langle u := \lambda a.C_a^{b,c}(bc)\rangle v\langle v := \lambda d.C_d^{e,f}(ef)\rangle\rangle \\
\longrightarrow_{(Var)} & z\langle x := \big(\lambda a.C_a^{b,c}(bc)\big)v\langle v := \lambda d.C_d^{e,f}(ef)\rangle\rangle \\
\longrightarrow_{(Var)} & z\langle x := \big(\lambda a.C_a^{b,c}(bc)\big)\big(\lambda d.C_d^{e,f}(ef)\big)\rangle
\end{array}
$$

We write $\longrightarrow_{\lambda\text{lxr}}^{*}$ to denote the reflexive closure of $\longrightarrow_{\lambda\text{lxr}}$.

We again stress that we are only looking at λlxr as an explicit substitution calculus here and we refer the reader to the original texts to get a full appreciation of its deeper connections to other fields of research.

1.3.1 Comparing λlxr and $'\Lambda$BIG

In λlxr, a weakening $W_x(M)$ states that x does not occur free in M. In $'\Lambda$BIG, we may consider weakenings as adding an idle name to a term. This is achieved by tensoring the bigraph $x : \epsilon \to \{x\}$ to the $'\Lambda$BIG term representing M. Similarly, a contraction $C_x^{y,z}(M)$ could be represented by a substitution bigraph $x/y \mid x/z$ composed with the $'\Lambda$BIG term representing M. These observations suggest the translation below from λlxr to $'\Lambda$BIG which we base on Milner's translation from Λ_{sub}.

$$
\begin{array}{rcl}
[\![x]\!]_{a,X\uplus x} & \stackrel{\text{def}}{=} & \text{var}_{ax} \oplus X \\
[\![\lambda x.M]\!]_{a,X} & \stackrel{\text{def}}{=} & (\text{lam}_{a(bx)} \oplus \text{id}_X)[\![M]\!]_{b,X\uplus x} \\
[\![MN]\!]_{a,X} & \stackrel{\text{def}}{=} & (\text{app}_{a(bc)} \oplus \text{id}_X)([\![M]\!]_{b,X} \mid [\![N]\!]_{c,X}) \\
[\![M\langle x := N\rangle]\!]_{a,X} & \stackrel{\text{def}}{=} & \big(\text{sub}_{a(bdx)} \oplus \text{id}_X\big)\big([\![M]\!]_{b,X\uplus x} \mid (\text{def}_{dx(c)} \oplus \text{id}_X)[\![N]\!]_{c,X}\big) \\
[\![W_x(M)]\!]_{a,X\uplus x} & \stackrel{\text{def}}{=} & (x \otimes \text{id}_a \oplus \text{id}_X)([\![M]\!]_{a,X}) \\
[\![C_x^{y,z}(M)]\!]_{a,X\uplus x} & \stackrel{\text{def}}{=} & (x/y \mid x/z \otimes \text{id}_a \oplus \text{id}_X)([\![M]\!]_{a,X\uplus\{y,z\}}).
\end{array}
$$

A $'\Lambda$BIG term arising from this translation may be considered to be linear as each occurrence of a variable control will be tagged with a different λ-name. For example, the λ-term $\lambda x.xx$ is translated as a (linear) λlxr term as $\lambda x.C_x^{y,z}(yz)$. The corresponding bigraph is then

$$\text{lam}_{a(bx)}(x/y \mid x/z \otimes \text{id}_{\{b\}})(\text{app}_{b(cd)} \oplus \text{id}_{\{y,z\}})(\text{var}_{cy} \oplus z \mid \text{var}_{dz} \oplus y)$$

which indeed looks linear. This bigraph may also be written (up to $\doteq$) as

$$\text{lam}_{a(bx)}(\text{app}_{b(cd)} \oplus \text{id}_{\{x\}})(\text{var}_{cx} \mid \text{var}_{dx})$$

which represents the original non-linear term. In fact, examining the two bigraph terms above, we may also write

$$\text{lam}_{a(bx)}(\text{app}_{b(cd)} \oplus \text{id}_{\{x\}})(\text{id}_1 \mid x/y \mid x/z)(\text{var}_{cy} \oplus z \mid \text{var}_{dz} \oplus y)$$

as an in-between term which has the 'contraction' pushed inside as far as possible.

Milner's presentation of the $\xrightarrow[\mathrm{C}]{}$ rule [Mil04] is in terms of replacing a variable y in M where there is a unique occurrence of y in M *i.e.* M is linear in y. This may suggest that the translation of λlxr terms above to 'linear' $'\Lambda_{\mathrm{BIG}}$ terms may make it easier (in comparison to the Λ_{sub} translation) to locate occurrences of redexes in a $'\Lambda_{\mathrm{BIG}}$ term.

If we consider the congruence axioms for λlxr terms and the associated translations, we notice that $\equiv_A$, $\equiv_{C1_c}$, $\equiv_{C2_c}$, $\equiv_{C_w}$, and $\equiv_{Cont2}$ follow immediately in $'\Lambda_{\mathrm{BIG}}$. The equivalence $\equiv_S$ which is used for composing substitutions does not follow from the conventional equivalences on bigraphs. To account for this in $'\Lambda_{\mathrm{BIG}}$, it may be possible to 1) define an equivalence relation on the bigraphs in $'\Lambda_{\mathrm{BIG}}$ such that a bigraph is related to another if they differ in the order of two adjacent substitutions and 2) define a bisimulation between one bigraph and a related bigraph. If we wished to simulate $\equiv_S$ *directly* in $'\Lambda_{\mathrm{BIG}}$, we would require two reaction rules to simulate both directions of the equivalence which would trivially lead to infinite reaction sequences and break PSN. We believe that wide substitution is a more natural method for $'\Lambda_{\mathrm{BIG}}$ to achieve full composition of substitutions.

One may consider replacing the rules of $'\Lambda_{\mathrm{BIG}}$ with reaction rules corresponding to those in λlxr rather than Λ_{sub}. To that end, we first observe that many reductions disappear through support equivalence. For example, using our translation, the variable x in the term $(y)(W_x(M))$ is carried to the outer interface and so $[\![(y)(W_x(M))]\!]_{a,X\uplus\{x,y\}} \simeq [\![W_x(yM)]\!]_{a,X\uplus\{x,y\}}$. This eliminates the need for rules similar to $(Weak2)$, $(WAbs)$, $(WApp1)$, $(WApp2)$, $(WSubs)$, and $(Cross)$ in the modified $'\Lambda_{\mathrm{BIG}}$. The rule $(Merge)$ becomes a casualty of composition. Similarly, rules $(CAbs)$, $(CApp1)$, $(CApp2)$, and $(CSubs)$ are no longer required. We then consider the remaining rules. The rule $(Weak1)$ would be similar to $\xrightarrow[\mathrm{D}]{}$ – the interface preservation is required by bigraph theory. The (B) rule already has its counterpart in $\xrightarrow[\mathrm{A}]{}$. This leaves the substitution propagation rules (Abs), $(App1)$, and $(App2)$, the local copy rule (Var), the composition rule $(Comp)$, and finally the duplication rule $(Cont1)$. The modified $'\Lambda_{\mathrm{BIG}}$ system would now resemble λxgc with composition (ignoring the linearity).

It would be interesting to see whether encoding this system as a Brs is possible or beneficial. The 'linearity' in the translation above seems forced and is 'lost' through composition (and hence equivalence of bigraphs). A specific contraction control may be necessary to properly model contractions. Also, as noted above, non-local substitution seems a better (or perhaps just more natural) way to achieve full composition of substitutions in a Brs. The use of distributive rules and local substitution may either require losing PSN or FCS using our translation.

Chapter 2

Proofs of confluence and PSN

2.1 Proof of confluence for Λ_{sub}

In this section we prove that the reduction relation in Λ_{sub} is confluent. The proof is based on Bloo and Rose's work on λxgc [BR95, Ros96a, Blo97]. This yields a proof of confluence for $'\Lambda\mathrm{BIG}$. The proofs are based on the correspondences between $\overline{\overline{\mathrm{gc}}}^{\twoheadrightarrow}$ and $\overline{\overline{\mathrm{D}}}^{\twoheadrightarrow}$ and between $\overline{\overline{\mathrm{x}}}^{\twoheadrightarrow}$ and $\overline{\overline{\mathrm{CD}}}^{\twoheadrightarrow}$.

Milner [Mil04] has already given a proof of weak confluence for Λ_{sub}. It follows from his bigraphical proof in $'\Lambda\mathrm{BIG}$. While we do not address his challenge of tackling (strong, open) confluence in the bigraphical setting, we prove (strong, closed) confluence for Λ_{sub} yielding a proof for $'\Lambda\mathrm{BIG}$. Direct, bigraphical proofs of strong confluence for $'\Lambda\mathrm{BIG}$ remain unpublished to date.

The proof of PSN in Section 2.2 is also based on Bloo and Rose's work and the reader may notice that some of our proofs are quite verbose in comparison. One reason is that in working with Λ_{sub}, we cannot use the inductive property that a distributive rule like $\overline{\mathrm{x}}^{\rightarrow}$ enjoys. On the other hand, some proofs may also be shortened due to the fact that substitution occurs 'at a distance'. In place of the inductive reasoning of Bloo and Rose, we employ contexts. Because of the existing verbosity, we have chosen to present the propositions and proofs in this section without much discussion. We refer the reader to Rose's tutorial [Ros96a] for more details as we have taken the proof structure directly from that work.

Notation. *We let $C[x]$ denote a context C where the hole is filled in with a variable x where this occurence of x is free in $C[x]$. We will sometimes annotate our proofs: the notation*

$$M \overset{3}{\equiv} N$$

indicates that the congruence can be shown by an application of Lemma 3.

Propositions 1. 1. $\overline{\mathrm{A}}^{\rightarrow}\mathrm{SN}$, 2. $\overline{\mathrm{A}}^{\rightarrow}\Diamond$, 3. $\overline{\mathrm{CD}}^{\rightarrow}\mathrm{SN}$, 4. $\overline{\mathrm{CD}}^{\rightarrow}\mathrm{CR}$, 5. $\overline{\mathrm{CD}}^{\rightarrow}\mathrm{UN}$

Proof.

1. Each Λx term has finitely many $\overline{\mathrm{A}}^{\rightarrow}$-redexes. Each $\overline{\mathrm{A}}^{\rightarrow}$-reduction decreases the number of $\overline{\mathrm{A}}^{\rightarrow}$-redexes. Hence, $\overline{\mathrm{A}}^{\rightarrow}\mathrm{SN}$.

2. $\overline{\mathrm{A}}^{\rightarrow} = \overline{\mathrm{b}}^{\rightarrow}$. Hence $\overline{\mathrm{A}}^{\rightarrow}\Diamond$ by [Ros96a, Proposition 1.1.10.2][1].

3. $\overline{\mathrm{CD}}^{\rightarrow}\mathrm{SN}$ is shown by finding a map $h : \Lambda x \to \mathbb{N}$ such that for all $M\ \overline{\mathrm{CD}}^{\rightarrow} N$ we have $h(M) > h(N)$. We call this map a weighting. Before we define the weighting, we introduce a labelling of terms. This labelling and proof of SN is adapted from [Bar84, Lemmas 11.12.17, 11.12.18].

   For a term M in Λx, we number the occurrences of variables (*i.e.* not the binders x in λx or $\langle x := N\rangle$) in M from the right to the left, depth-first, according to the abstract syntax tree, starting with the number 0. Give the n^{th} occurrence the index 2^n. e.g.

   $$xy((\lambda z.x)\langle x := wv\rangle) \text{ becomes } x^{16}y^{8}((\lambda z.x^{4})\langle x := w^{2}v^{1}\rangle).$$

[1] A bigraphical proof would consist of a case split over the possible overlappings. There are two non-trivial possibilities. Either both redexes are independent or one lies entirely within the other. The first case yields the diamond property [Mil04]. It should be trivial to prove the same for the second case using Milner's theorems in that text.

We define the weighting as: $h(x^n) = n, h(MN) = h(M) + h(N), h(\lambda x.M) = h(M), h(M\langle x := N\rangle) = h(M) + h(N)$ *i.e.* $h(M) =$ the sum of all indices in M.

Next we state two properties on labelled terms

$$\begin{aligned} Prp(M) &\stackrel{\text{def}}{=} \forall x^i \subseteq M, \text{ if } x^i \text{ is bound by } \langle x := P\rangle \text{ then } i > h(P), \\ PrpH(M) &\stackrel{\text{def}}{=} h(P) > 0 \text{ for all subterms } P \text{ of } M. \end{aligned}$$

We have the following properties.

(a) If $Prp(M)$ and $PrpH(M)$ then if $M \xrightarrow[\text{CD}]{} M'$, $h(M) > h(M')$

(b) $Prp(M)$ is preserved through $\xrightarrow[\text{CD}]{}$ reduction:

- In the $\xrightarrow[\text{D}]{}$ case, the discarded substitution binds no variables.
- For the $\xrightarrow[\text{C}]{}$ case, consider

$$M \equiv C_1[C_2[x]\langle x := P\rangle] \xrightarrow[\text{C}]{} C_1[C_2[P]\langle x := P\rangle] \equiv M'.$$

We only need consider the variables in the new copy of P – the proof follows by $PrpM$. These variables are either free in M' or else bound by some abstraction or substitution above $C_2[P]\langle x := P\rangle$ as variable capture does not occur. As $Prp(M)$, these variables satisfy the necessary condition and so $Prp(M')$.

(c) Proving $PrpH(M)$ amounts to proving that all variables have a positive, non-zero label. Thus, $PrpH(M)$ is preserved through $\xrightarrow[\text{CD}]{}$ reduction.

(d) Label any term M with the initial labelling described above. We have $Prp(M)$ and $PrpH(M)$ and so if $M \xrightarrow[\text{CD}]{\gg} M'$, $h(M) > h(M')$.

As $h(M)$ is finite for all terms M, the proof then follows.

4. $\xrightarrow[\text{CD}]{}$LC can be proved by inspecting the cases of the proof of $\xrightarrow[\text{ACD}]{}$LC given in [Mil04, Propositions 5.8, 5.5]. $\xrightarrow[\text{CD}]{}$SN and $\xrightarrow[\text{CD}]{}$LC imply $\xrightarrow[\text{CD}]{}$CR by Newman's lemma [New42].

5. $\xrightarrow[\text{CD}]{}$CR implies $\xrightarrow[\text{CD}]{}$UN [Ter03, Theorem 1.2.2(i), p. 17].

□

We skipped past the proof of $\xrightarrow[\text{CD}]{}$LC above so we will briefly outline the important interactions between the rules. As the reduction $\xrightarrow[\text{C}]{}$ is wide, the interactions are not similar to λxgc where the critical pairs can be determined as usual in term rewriting systems. Milner has classified the ways in which bigraphical rules may overlap and has provided local confluence theorems for those cases [Mil04]. The three interesting cases for Λ_{sub} are as follows.

1. A $\xrightarrow[\text{C}]{}$ or $\xrightarrow[\text{D}]{}$ redex lies totally under another $\xrightarrow[\text{C}]{}$ or $\xrightarrow[\text{D}]{}$ redex.

2. The most interesting case for Λ_{sub} is when a redex is partly inside another. This occurs due to the wide reaction rules that BRSs allow. $\xrightarrow[\text{C}]{}$ is a wide rule so it is possible that the variable x to be replaced is not a sibling of the substitution definition $x := N$ – in fact this is only the case in terms of the form $x\langle x := N\rangle$. Generally, x is a subterm of some term M. The case occurs when x lies under a $\xrightarrow[\text{CD}]{}$ redex in M, *e.g.*:

$$\big(P(\underline{(\lambda y.x)\langle y := N\rangle})\big)\langle x := N\rangle, \quad y \neq x$$

where a free x lies under the underlined $\xrightarrow[\text{D}]{}$ redex.

3. The last case is when two redexes partly overlap. A term $M\langle x := N\rangle$ cannot be both a $\xrightarrow[\text{C}]{}$ index and a $\xrightarrow[\text{D}]{}$ index as either $x \in \text{fv}(M)$ or not. Thus, a $\xrightarrow[\text{C}]{}$ redex cannot partly overlap with a $\xrightarrow[\text{D}]{}$ redex. Two $\xrightarrow[\text{D}]{}$ redexes which overlap must be the same redex. The only case of partly overlapping redexes is when two $\xrightarrow[\text{C}]{}$ redexes overlap and in this case, the overlap must be on the substitution definition $x := N$. Milner provides a general bigraphical proof for this situation. The essential property of the $\xrightarrow[\text{C}]{}$ reduction is that

the substitution definition $x := N$ remains after a $\xrightarrow[\mathrm{C}]{}$ reduction. For Λ_{sub}, the following diagram explains the resolution.

$$\begin{array}{ccc}
(\cdots x \cdots x \cdots)\langle x := N\rangle & \xrightarrow{\mathrm{C}} & (\cdots N \cdots x \cdots)\langle x := N\rangle \\
{\scriptstyle\mathrm{C}}\downarrow & & \downarrow{\scriptstyle\mathrm{C}} \\
(\cdots x \cdots N \cdots)\langle x := N\rangle & \xrightarrow[\mathrm{C}]{} & (\cdots N \cdots N \cdots)\langle x := N\rangle
\end{array}$$

Notation. *As in [Ros96a], we denote the unique* $\xrightarrow[\mathrm{CD}]{}$*-normal form of* M *as* $\downarrow_{\mathrm{CD}}(M)$*. We say that* M *is* pure *if* $M \equiv \downarrow_{\mathrm{CD}}(M) \in \Lambda$*. We denote the unique* $\xrightarrow[\mathrm{D}]{}$*-normal form of* M *as* $\downarrow_{\mathrm{D}}(M)$*. We say that* M *is* garbage-free *if* $M \equiv \downarrow_{\mathrm{D}}(M)$.

The following lemmas concerning unique normal forms will be useful. They are required in some proofs where we cannot avail of the inductive methods that were originally employed.

Lemmas 2 (normal forms). *For all* Λ_{sub} *terms* P *and* Q*,*

1. $\downarrow_{\mathrm{CD}}(PQ) \equiv \downarrow_{\mathrm{CD}}(P)\ \downarrow_{\mathrm{CD}}(Q)$
2. $\downarrow_{\mathrm{CD}}(\lambda x.P) \equiv \lambda x.\ \downarrow_{\mathrm{CD}}(P)$
3. $\downarrow_{\mathrm{CD}}((PQ)\langle x_1 := N_1\rangle \cdots \langle x_n := N_n\rangle)$
   $\equiv \downarrow_{\mathrm{CD}}(P\langle x_1 := N_1\rangle \cdots \langle x_n := N_n\rangle)\ \downarrow_{\mathrm{CD}}(Q\langle x_1 := N_1\rangle \cdots \langle x_n := N_n\rangle)$
4. $\downarrow_{\mathrm{CD}}\big((\lambda x.P)\langle x_1 := N_1\rangle \cdots \langle x_n := N_n\rangle\big)$
   $\equiv \downarrow_{\mathrm{CD}}(\lambda x.P\langle x_1 := N_1\rangle \cdots \langle x_n := N_n\rangle)$
   $\equiv \lambda x.\ \downarrow_{\mathrm{CD}}(P\langle x_1 := N_1\rangle \cdots \langle x_n := N_n\rangle)$
5. *1,2, and 4 hold with* $\downarrow_{\mathrm{CD}}$ *replaced by* $\downarrow_{\mathrm{D}}$

Proof. We use the fact that reductions in Λ_{sub} may be applied to any subterm of a term (this is realised in $'\Lambda_{\mathrm{BIG}}$ by the fact that all controls besides the atomic var control are active). Specifically; for (1), P and Q can not interact via $\xrightarrow[\mathrm{CD}]{}$ reductions. (1) then follows as $\xrightarrow[\mathrm{CD}]{}$UN. (4) follows by the variable convention ($x \neq x_i, 1 \leq 1 \leq n$) and then an application of (2).

□

Lemma 3 (representation). *For all terms* M, N *and variable* x*,*

$$\downarrow_{\mathrm{CD}}(M\langle x := N\rangle) \equiv \downarrow_{\mathrm{CD}}(M)[x := \downarrow_{\mathrm{CD}}(N)]$$

Proof. We induct over the number of symbols in $M, N_1, \ldots N_n$ and show that

$$\downarrow_{\mathrm{CD}}(M\langle x_1 := N_1\rangle \cdots \langle x_n := N_n\rangle) \equiv \downarrow_{\mathrm{CD}}(M)[x_1 := \downarrow_{\mathrm{CD}}(N_1)] \cdots [x_n := \downarrow_{\mathrm{CD}}(N_n)]$$

We break the proof over the structure of M:

Case $M \equiv x, n = 0$: $\quad \downarrow_{\mathrm{CD}}(x) \equiv \downarrow_{\mathrm{CD}}(x)$

Case $M \equiv x, n > 0$:

if $x \neq x_1$ then

$$\begin{array}{rl}
 & \downarrow_{\mathrm{CD}}(x\langle x_1 := N_1\rangle \cdots \langle x_n := N_n\rangle) \\
\equiv & \downarrow_{\mathrm{CD}}(x\langle x_2 := N_2\rangle \cdots \langle x_n := N_n\rangle) \\
\overset{\mathrm{IH}}{\equiv} & \downarrow_{\mathrm{CD}}(x)[x_2 := \downarrow_{\mathrm{CD}}(N_2)] \cdots [x_n := \downarrow_{\mathrm{CD}}(N_n) \\
\equiv & \downarrow_{\mathrm{CD}}(x)[x_1 := \downarrow_{\mathrm{CD}}(N_1)] \cdots [x_n := \downarrow_{\mathrm{CD}}(N_n)
\end{array}$$

if $x = x_1$ then

$$\begin{array}{rc} & \downarrow_{\mathrm{CD}}(x\langle x_1 := N_1\rangle \cdots \langle x_n := N_n\rangle) \\ \equiv & \downarrow_{\mathrm{CD}}(N_1\langle x_2 := N_2\rangle \cdots \langle x_n := N_n\rangle) \\ \overset{\mathrm{IH}}{\equiv} & \downarrow_{\mathrm{CD}}(N_1)[x_2 :=\downarrow_{\mathrm{CD}}(N_2)] \cdots [x_n :=\downarrow_{\mathrm{CD}}(N_n) \\ \equiv & x[x :=\downarrow_{\mathrm{CD}}(N_1)][x_2 :=\downarrow_{\mathrm{CD}}(N_2)] \cdots [x_n :=\downarrow_{\mathrm{CD}}(N_n) \\ \equiv & \downarrow_{\mathrm{CD}}(x)[x_1 :=\downarrow_{\mathrm{CD}}(N_1)] \cdots [x_n :=\downarrow_{\mathrm{CD}}(N_n) \end{array}$$

Case $M \equiv PQ$

$$\begin{array}{rc} & \downarrow_{\mathrm{CD}}(M\langle x_1 := N_1\rangle \cdots \langle x_n := N_n\rangle) \\ \equiv & \downarrow_{\mathrm{CD}}((PQ)\langle x_1 := N_1\rangle \cdots \langle x_n := N_n\rangle) \\ \overset{2.3}{\equiv} & \downarrow_{\mathrm{CD}}(P\langle x_1 := N_1\rangle \cdots \langle x_n := N_n\rangle)\ \downarrow_{\mathrm{CD}}(Q\langle x_1 := N_1\rangle \cdots \langle x_n := N_n\rangle) \\ \overset{\mathrm{IH}}{\equiv} & \big(\downarrow_{\mathrm{CD}}(P)[x_1 :=\downarrow_{\mathrm{CD}}(N_1)] \cdots [x_n :=\downarrow_{\mathrm{CD}}(N_n)] \\ & \downarrow_{\mathrm{CD}}(Q)[x_1 :=\downarrow_{\mathrm{CD}}(N_1)] \cdots [x_n :=\downarrow_{\mathrm{CD}}(N_n)]\big) \\ \equiv & (\downarrow_{\mathrm{CD}}(P)\ \downarrow_{\mathrm{CD}}(Q))[x_1 :=\downarrow_{\mathrm{CD}}(N_1)] \cdots [x_n :=\downarrow_{\mathrm{CD}}(N_n)] \\ \overset{2.1}{\equiv} & \downarrow_{\mathrm{CD}}(PQ)[x_1 :=\downarrow_{\mathrm{CD}}(N_1)] \cdots [x_n :=\downarrow_{\mathrm{CD}}(N_n)]) \end{array}$$

Case $M \equiv \lambda x.P$

$$\begin{array}{rc} & \downarrow_{\mathrm{CD}}(M\langle x_1 := N_1\rangle \cdots \langle x_n := N_n\rangle) \\ \equiv & \downarrow_{\mathrm{CD}}((\lambda x.P)\langle x_1 := N_1\rangle \cdots \langle x_n := N_n\rangle) \\ \overset{2.4}{\equiv} & \lambda x.\ \downarrow_{\mathrm{CD}}(P\langle x_1 := N_1\rangle \cdots \langle x_n := N_n\rangle) \\ \overset{\mathrm{IH}}{\equiv} & \lambda x.\ \downarrow_{\mathrm{CD}}(P)[x_1 :=\downarrow_{\mathrm{CD}}(N_1)] \cdots [x_1 :=\downarrow_{\mathrm{CD}}(N_n)] \\ & (x \neq x_i, 1 \leq i \leq n \text{ by the variable convention}) \\ \equiv & \downarrow_{\mathrm{CD}}(\lambda x.P)[x_1 :=\downarrow_{\mathrm{CD}}(N_1)] \cdots [x_n :=\downarrow_{\mathrm{CD}}(N_n)] \end{array}$$

Case $M \equiv P\langle y := Q\rangle$[2]

$$\begin{array}{rc} & \downarrow_{\mathrm{CD}}(M\langle x_1 := N_1\rangle \cdots \langle x_n := N_n\rangle) \\ \equiv & \downarrow_{\mathrm{CD}}(P\langle y := Q\rangle\langle x_1 := N_1\rangle \cdots \langle x_n := N_n\rangle) \\ & \text{(inductive case over the symbols in } P, Q, N_1 \ldots N_n \\ & \text{which has less symbols than } P\langle y := Q\rangle, N_1 \ldots N_n) \\ \overset{\mathrm{IH}}{\equiv} & \downarrow_{\mathrm{CD}}(P)[y :=\downarrow_{\mathrm{CD}}(Q)][x_1 :=\downarrow_{\mathrm{CD}}(N_1)] \cdots [x_n :=\downarrow_{\mathrm{CD}}(N_n)]) \\ & \text{(use IH backwards over } \downarrow_{\mathrm{CD}}(P)[y :=\downarrow_{\mathrm{CD}}(Q)] \\ & \text{which has less symbols than } P, Q, N_1 \ldots N_n) \\ \overset{\mathrm{IH}}{\equiv} & \downarrow_{\mathrm{CD}}(P\langle y := Q\rangle)[x_1 :=\downarrow_{\mathrm{CD}}(N_1)] \cdots [x_n :=\downarrow_{\mathrm{CD}}(N_n)]) \end{array}$$

□

Corollary 4 (substitution lemma [Ros96a]).

$$M\langle x := N\rangle\langle y := P\rangle \overset{\ll}{\underset{\mathrm{CD}}{\longleftrightarrow}}\!\!\gg M\langle y := P\rangle\langle x := N\langle y := P\rangle\rangle$$

Proof. Follows from previous lemma, the λ-calculus substitution lemma

$$M[x := N][y := P] \equiv M[y := P][x := N[y := P]] \quad (*)\,,$$

[2]Thanks to Thomas Hildebrandt for explaining this case to me.

and the fact that $\downarrow_{\mathrm{CD}}(Q)$ is a pure term for any Q. Explicitly;

$$\begin{array}{rc} & \downarrow_{\mathrm{CD}}(M\langle x := N\rangle\langle y := P\rangle) \\ \overset{3}{\equiv} & \downarrow_{\mathrm{CD}}(M)[x :=\downarrow_{\mathrm{CD}}(N)][y :=\downarrow_{\mathrm{CD}}(P)] \\ \overset{(*)}{\equiv} & \downarrow_{\mathrm{CD}}(M)[y :=\downarrow_{\mathrm{CD}}(P)][x :=\downarrow_{\mathrm{CD}}(N)[y :=\downarrow_{\mathrm{CD}}(P)]] \\ \overset{3}{\equiv} & \downarrow_{\mathrm{CD}}(M)[y :=\downarrow_{\mathrm{CD}}(P)][x :=\downarrow_{\mathrm{CD}}(N\langle y := P\rangle)] \\ \overset{3}{\equiv} & \downarrow_{\mathrm{CD}}(M\langle y := P\rangle\langle x := N\langle y := P\rangle\rangle \end{array}$$

and the corollary follows. □

The following is slightly different to the corresponding proof for $\lambda\mathrm{xgc}$. In $\Lambda_{\mathtt{sub}}$, the only type of reduction which reduces the set of free variables of a term is $\xrightarrow[\mathrm{D}]{}$. In $\lambda\mathrm{xgc}$, $\xrightarrow[\mathrm{gc}]{}$ is not the only reduction with this property – $\xrightarrow[\mathrm{xvgc}]{}$ (a subrelation of $\xrightarrow[\mathrm{x}]{}$) may also reduce the set of free variables. The statement $\downarrow_{\mathrm{CD}}(M) \equiv\downarrow_{\mathrm{CD}}(N)$ in the first two propositions follows trivially by $\xrightarrow[\mathrm{CD}]{}$UN.

Propositions 5. *For any M,*

1. *If $M \xrightarrow[\mathrm{D}]{} N$ then* $\mathrm{fv}(M) \supseteq \mathrm{fv}(N)$ *and* $\downarrow_{\mathrm{CD}}(M) \equiv\downarrow_{\mathrm{CD}}(N)$.
2. *If $M \xrightarrow[\mathrm{C}]{} N$ then* $\mathrm{fv}(M) = \mathrm{fv}(N)$ *and* $\downarrow_{\mathrm{CD}}(M) \equiv\downarrow_{\mathrm{CD}}(N)$.
3. *If $M \xrightarrow[\mathrm{A}]{} N$ then* $\mathrm{fv}(M) = \mathrm{fv}(N)$
4. *If M is garbage-free then* $\mathrm{fv}(M) = \mathrm{fv}(\downarrow_{\mathrm{CD}}(M))$.

Proof.

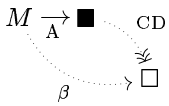
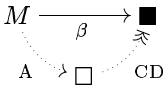
1. $\xrightarrow[\mathrm{D}]{}$ may discard some free variables, hence $\mathrm{fv}(M) \supseteq \mathrm{fv}(N)$.
2. Let $M \equiv C'\big[C[x]\langle x := Q\rangle\big] \xrightarrow[\mathrm{C}]{} C'\big[C[Q]\langle x := Q\rangle\big] \equiv N$. The occurrence of x which is replaced was bound in M so no free variables are lost. Any y which is bound in Q in M is bound within Q or by some binder above Q in M. The same is true of the copy of Q in N and so the set of free variables does not increase and $\mathrm{fv}(M) = \mathrm{fv}(N)$.
3. By definition, $(\lambda x.P)Q$ has the same set of free variables as $P\langle x := Q\rangle$.
4. There is a reduction path

$$M \equiv M'_0 \xrightarrow[\mathrm{CD}]{} M_1 \xrightarrow[\mathrm{D}]{=} M'_1 \xrightarrow[\mathrm{CD}]{} M_2 \xrightarrow[\mathrm{D}]{=} M'_2 \xrightarrow[\mathrm{CD}]{} \cdots \xrightarrow[\mathrm{D}]{=} M'_n \equiv\downarrow_{\mathrm{CD}}(M)$$

where

- $M_{i+1} \equiv M'_{i+1}$ if $M'_i \xrightarrow[\mathrm{CD}]{} M_{i+1}$ is a $\xrightarrow[\mathrm{C}]{}$ step which does not substitute for the last free occurrence of a variable *i.e.* it does not create garbage,
- $M_{i+1} \xrightarrow[\mathrm{D}]{} M'_{i+1}$ if $M'_i \xrightarrow[\mathrm{CD}]{} M_{i+1}$ is a $\xrightarrow[\mathrm{C}]{}$ step which does substitute for the last free occurrence of a variable *i.e.* it creates garbage,
- all M'_i are garbage free.

We now show that $\mathrm{fv}(M'_i) = \mathrm{fv}(M'_{i+1})$. For the first case of $\xrightarrow[\mathrm{C}]{}$ reduction, this follows by (2) above. For the second case, the $\xrightarrow[\mathrm{D}]{}$ discards some free variables but those variables were copied by the $\xrightarrow[\mathrm{C}]{}$ reduction and avoided variable capture. The proof follows by $\xrightarrow[\mathrm{CD}]{}$UN. □

Proposition 6. *For pure M, 1.*  *and 2.* 

Proof. There exists a pure context C such that $M \equiv C[(\lambda x.P)Q]$ with P and Q pure *i.e.* $P \equiv{\downarrow_{\mathrm{CD}}}(P), Q \equiv{\downarrow_{\mathrm{CD}}}(Q)$.

1. We have

$$\begin{array}{rc} & C[(\lambda x.P)Q] \\ \xrightarrow[\mathrm{A}]{} & C[P\langle x := Q\rangle] \\ \xrightarrow[\mathrm{CD}]{} & C[{\downarrow_{\mathrm{CD}}}(P\langle x := Q\rangle)] \\ \overset{3}{\equiv} & C[{\downarrow_{\mathrm{CD}}}(P)[x :={\downarrow_{\mathrm{CD}}}(Q)])] \\ \equiv & C[P[x := Q]] \end{array}$$

   and $C[(\lambda x.P)Q] \xrightarrow[\beta]{} C[P[x := Q]]$.

2. Follows similarly.

□

Lemmas 7 (Projection).

1. *For all Λ_{sub}-terms M,*
$$\begin{array}{ccc} M & \xrightarrow{\mathrm{A}} & N \\ \downarrow_{\mathrm{CD}} & & \downarrow_{\mathrm{CD}} \\ {\downarrow_{\mathrm{CD}}}(M) & \overset{}{\underset{\beta}{\twoheadrightarrow}} & {\downarrow_{\mathrm{CD}}}(N) \end{array}$$
2. *For garbage-free M,*
$$\begin{array}{ccc} M & \xrightarrow{\mathrm{A}} & N \\ \downarrow_{\mathrm{CD}} & & \downarrow_{\mathrm{CD}} \\ {\downarrow_{\mathrm{CD}}}(M) & \overset{+}{\underset{\beta}{\longrightarrow}} & {\downarrow_{\mathrm{CD}}}(N) \end{array}$$

Proof. We prove 2 by inducting over the structure of M:

Case $M \equiv (\lambda x.P)Q, N \equiv P\langle x := Q\rangle$:

$$\begin{array}{rc} & {\downarrow_{\mathrm{CD}}}(M) \\ \overset{2}{\equiv} & (\lambda x.\, {\downarrow_{\mathrm{CD}}}(P))\, {\downarrow_{\mathrm{CD}}}(Q) \\ \xrightarrow[\beta]{} & {\downarrow_{\mathrm{CD}}}(P)[x :={\downarrow_{\mathrm{CD}}}(Q)] \\ \overset{3}{\equiv} & {\downarrow_{\mathrm{CD}}}(P\langle x := Q\rangle) \end{array}$$

Case $M \equiv PQ, N \equiv P'Q$

$${\downarrow_{\mathrm{CD}}}(M) \overset{2.1}{\equiv} {\downarrow_{\mathrm{CD}}}(P)\, {\downarrow_{\mathrm{CD}}}(Q) \xrightarrow[\beta]{+}{}^{\mathrm{IH}}\ {\downarrow_{\mathrm{CD}}}(P')\, {\downarrow_{\mathrm{CD}}}(Q) \overset{2.1}{\equiv} {\downarrow_{\mathrm{CD}}}(N)$$

Case $M \equiv PQ, N \equiv PQ'$: As above

Case $M \equiv \lambda x.P, N \equiv \lambda x.P'$

$${\downarrow_{\mathrm{CD}}}(M) \equiv {\downarrow_{\mathrm{CD}}}(\lambda x.P) \overset{2.2}{\equiv} \lambda x.\, {\downarrow_{\mathrm{CD}}}(P) \xrightarrow[\beta]{+}{}^{\mathrm{IH}}\ \lambda x.\, {\downarrow_{\mathrm{CD}}}(P') \overset{2.2}{\equiv} {\downarrow_{\mathrm{CD}}}(\lambda x.P')$$

Case $M \equiv P\langle x := Q\rangle, N \equiv P'\langle x := Q\rangle$: Similar to the next case.

Case $M \equiv P\langle x := Q\rangle, N \equiv P\langle x := Q'\rangle$

By the inductive hypothesis, we know that $\downarrow_{\mathrm{CD}}(Q) \xrightarrow[\beta]{+} \downarrow_{\mathrm{CD}}(Q')$ and so

$$\downarrow_{\mathrm{CD}}(P)[x :=\downarrow_{\mathrm{CD}}(Q)] \xrightarrow[\beta]{+} \downarrow_{\mathrm{CD}}(P)[x :=\downarrow_{\mathrm{CD}}(Q')].$$

since by the fact that M is garbage free we have that $x \in \mathrm{fv}(P)$ and by Proposition 5.4, it follows that $x \in \mathrm{fv}(\downarrow_{\mathrm{CD}}(P))$.

Now, by application of Lemma 3 twice we have

$$\downarrow_{\mathrm{CD}}(M) \overset{3}{\equiv} \downarrow_{\mathrm{CD}}(P)[x :=\downarrow_{\mathrm{CD}}(Q)] \xrightarrow[\beta]{+} \downarrow_{\mathrm{CD}}(P)[x :=\downarrow_{\mathrm{CD}}(Q')] \overset{3}{\equiv} \downarrow_{\mathrm{CD}}(N).$$

The proof of 1 is similar except that we do not know in the last case that $x \in \mathrm{fv}(\downarrow_{\mathrm{CD}}(P))$ and so we have to use $\xrightarrow[\beta]{}$, allowing the reflexive closure to provide identity ($\equiv$) *i.e.* we do a case split taking either $x \in \mathrm{fv}(P)$ (proof as above) or $x \notin \mathrm{fv}(P)$ (in which case $\downarrow_{\mathrm{CD}}(M) \equiv\downarrow_{\mathrm{CD}}(N)$). □

Proposition 6, Lemma 7, and $\xrightarrow[\mathrm{CD}]{}$UN prove that $\Lambda_{\mathtt{sub}}$ is a conservative extension of the $\lambda\beta$-calculus, with $\xrightarrow[\mathrm{CD}]{}$ as translation.

Theorem 8. *For pure terms M, N: $M \xrightarrow[\mathrm{ACD}]{} N \iff M \xrightarrow[\beta]{} N$*

Proof.

Case $\Leftarrow$: Assume $M \xrightarrow[\beta]{} N$; we then prove by induction on the length n of the β-reduction, using Proposition 6.2 in each step:

Case: $n = 0$ Trivial

Inductive hypothesis (IH): $n = k \;\; M \xrightarrow[\beta]{k} N \Longrightarrow M \xrightarrow[\mathrm{ACD}]{} N$

Case: $n = k + 1$

$$M \xrightarrow[\beta]{k+1} N \Rightarrow M \xrightarrow[\beta]{} M' \xrightarrow[\beta]{k} N$$

By Proposition 6.2 and IH, we have $M \xrightarrow[\mathrm{A}]{} \xrightarrow[\mathrm{CD}]{} M' \xrightarrow[\mathrm{ACD}]{} N$.

Case $\Rightarrow$: We induct over the length of the $\xrightarrow[\mathrm{ACD}]{}$-reduction to prove

$$\begin{array}{ccccccccc} M & \xrightarrow{\mathrm{ACD}} & M_1 & \xrightarrow{\mathrm{ACD}} & \cdots & \xrightarrow{\mathrm{ACD}} & M_{n-1} & \xrightarrow{\mathrm{ACD}} & N \\ \| & & \downarrow{\scriptstyle \mathrm{CD}} & & & & \downarrow{\scriptstyle \mathrm{CD}} & & \| \\ M & \xrightarrow[\beta]{} & \downarrow_{\mathrm{CD}}(M_1) & \xrightarrow[\beta]{} & \cdots & \xrightarrow[\beta]{} & \downarrow_{\mathrm{CD}}(M_{n-1}) & \xrightarrow[\beta]{} & N \end{array}$$

Each step in the top of each square above is one of $\xrightarrow[\mathrm{D}]{}$, $\xrightarrow[\mathrm{C}]{}$, or $\xrightarrow[\mathrm{A}]{}$ and so each square respectively gives rise to one of the following diagrams:

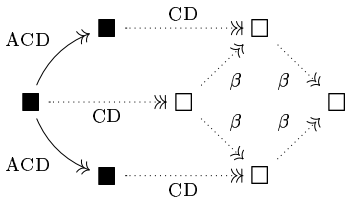
$$\begin{array}{ccc} M_i & \xrightarrow{\mathrm{CD}} & M_{i+1} \\ \downarrow{\scriptstyle \mathrm{CD}} & & \downarrow{\scriptstyle \mathrm{CD}} \\ \downarrow_{\mathrm{CD}}(M_i) & = & \downarrow_{\mathrm{CD}}(M_{i+1}) \end{array} \qquad \begin{array}{ccc} M_i & \xrightarrow{\mathrm{A}} & M_{i+1} \\ \downarrow{\scriptstyle \mathrm{CD}} & & \downarrow{\scriptstyle \mathrm{CD}} \\ \downarrow_{\mathrm{CD}}(M_i) & \xrightarrow[\beta]{} & \downarrow_{\mathrm{CD}}(M_{i+1}) \end{array}$$

which follow respectively by $\xrightarrow[\mathrm{CD}]{}$UN and Lemma 7.1.

□

Corollary 9. $\overrightarrow{\text{ACD}}\,\text{CR}$

Proof. We prove the following diagrammatic proposition which states that $\overrightarrow{\text{ACD}}$ is strongly confluent. Strong confluence implies CR.

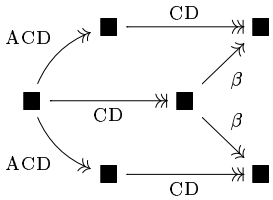


We have shown that

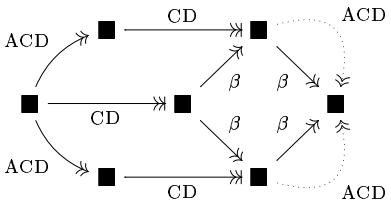
1. $\overrightarrow{\beta} \subseteq \overrightarrow{\text{ACD}}^{\gg}$ (by Proposition 6.2),
2. $\forall M \in \Lambda_{\text{sub}} : M \overrightarrow{\text{ACD}}^{\gg} \downarrow_{\text{CD}}(M) \in \Lambda$,
3. $\forall M, N \in \Lambda_{\text{sub}} : M \overrightarrow{\text{ACD}} N \Rightarrow \downarrow_{\text{CD}}(M) \overrightarrow{\beta}^{\gg} \downarrow_{\text{CD}}(N)$. This follows from the fact that a $M \overrightarrow{\text{ACD}} N$ reduction is either a $M \overrightarrow{\text{A}} N$ reduction or a $M \overrightarrow{\text{CD}} N$ reduction. The former case is shown by Lemma 7.1 and the latter case from $\overrightarrow{\text{CD}}\,\text{UN}$.

The proof follows from the generalised interpretation method (GIM) [KR97] which was inspired by Hardin's interpretation method [Har89]. In the notation of [KR97], A is the set of terms in Λ_{sub}, B is the set of pure λ-terms, $R = \overrightarrow{\text{ACD}}$, $R' = \overrightarrow{\beta}$ and $f = \overrightarrow{\text{CD}}^{\gg}$.

Explicitly; from 2 we can fill in the horizontal arrows below and by 3 we can fill in the diagonal β arrows below.



By the confluence of $\overrightarrow{\beta}$ we have



where, by 1, the dotted arrows can be filled in. Hence, $\overrightarrow{\text{ACD}}\,\text{CR}$. □

Corollary 10 (closed confluence in $'\Lambda_{\text{BIG}}$). *Confluence holds for every image of a Λ_{sub}-term without metavariables in $'\Lambda_{\text{BIG}}$.*

Proof. The result follows from Corollary 9 and [Mil04, Proposition 5.5]. □

2.2 An inductive proof of PSN

In this section, we prove that $\overrightarrow{\mathrm{ACD}}$ preserves strong normalisation (PSN) of β-reduction. PSN means that if $M \in \Lambda$ is strongly normalising for β-reduction then it is strongly normalising for $\overrightarrow{\mathrm{ACD}}$ reduction.

The proofs are based on Bloo and Rose's work [BR95, Ros96a, Blo97] on methods of proving PSN for calculi with explicit substitutions. We follow the inductive proof in [BR95], employing the technique of garbage-free reduction to assist us. Bloo [Blo97] gives an alternative inductive proof and Bloo and Geuvers [Blo97, BG99] use the recursive path ordering (RPO) technique. Our inductive proof is inelegant and we present a better proof in Section 2.4.4. We have not tried a proof using the RPO technique as we felt that the same complications may occur as in the following inductive proof. We thought specifically that the fact that the substitution definition persists after a $\overrightarrow{\mathrm{C}}$ reduction and remains in place may be a complication for an RPO proof.

In Section 2.2.1, we follow Rose and prove PSN for a calculus $\Lambda_{\mathrm{sub}\downarrow\mathrm{D}}$ where reduction, called garbage-free reduction, can be described as 'do an $\overrightarrow{\mathrm{A}}$ or a $\overrightarrow{\mathrm{C}}$ followed by total garbage collection.'

Section 2.2.2 introduces two new calculi $\Lambda_{\mathrm{sub}\mathbb{G}}$ and $\Lambda_{\mathrm{subC}^\flat}$. They are both weak versions of Λ_{sub} in that the reduction relation is a subset of $\overrightarrow{\mathrm{ACD}}$, respectively omitting all or some copying between substitutions. This copying between substitutions effectively *is* compositions of substitutions in Λ_{sub}. We prove PSN for $\Lambda_{\mathrm{subC}^\flat}$ in this section by adapting Rose's inductive proofs for $\lambda\mathrm{xgc}$. We show that as $\Lambda_{\mathrm{subC}^\flat}$ allows some form of composition of substitutions and $\lambda\mathrm{xgc}$ does not[3], the set of strongly normalising terms of $\Lambda_{\mathrm{subC}^\flat}$ is a subset of the set of strongly normalising terms of $\lambda\mathrm{xgc}$.

Section 2.2.3 discusses why the property subSN used in the proof of PSN for $\Lambda_{\mathrm{subC}^\flat}$ is not a sufficient property for reasoning about infinite reduction paths inside garbage in Λ_{sub}. As might be expected, the problem is with copying between substitutions. We show that these reductions (that we disallowed in $\Lambda_{\mathrm{sub}\mathbb{G}}$) conspire to make the set of strongly normalising terms of Λ_{sub} a strict subset of that of both $\lambda\mathrm{xgc}$ and $\Lambda_{\mathrm{sub}\mathbb{G}}$. The issue is that more cases of infinite reductions inside garbage can occur. We identify the set of strongly normalising terms of Λ_{sub} but do not have a neat characterisation as in $\Lambda_{\mathrm{subC}^\flat}$ or $\lambda\mathrm{xgc}$.

Section 2.2.4 finally tackles the proof of PSN for Λ_{sub}.

2.2.1 PSN for $\Lambda_{\mathrm{sub}\downarrow\mathrm{D}}$

Definition (body of a substitution [Ros96a]). *We say N is a* body of a substitution in M *if for some P, x, $P\langle x := N\rangle$ is a subterm of M.*

Notation (in(side), under a body of substitution). *We say N is* in a body of substitution P *if N is a subterm of P. We say N is* under a body of substitution P *if N is a subterm of M in $M\langle x := P\rangle$.*

Definition (top-level substitution). *If a substitution $\langle x := P\rangle$ in a term M does not lie inside any other substitution then it is called a* top-level substitution. *Top-level substitutions may lie* under *other substitutions.*

Definition (garbage-free reduction). $\overrightarrow{\mathrm{AC}\downarrow\mathrm{D}}$ *is* $(\overrightarrow{\mathrm{AC}} \cdot \overrightarrow{\mathrm{D}}^{\gg})$*, i.e., the union of the composition of* $\overrightarrow{\mathrm{AC}}$ *with complete garbage collection. We denote the garbage-free reduction calculus* $\Lambda_{\mathrm{sub}\downarrow\mathrm{D}}$.

Garbage-free reduction is reduction "where all garbage is removed as soon as possible"[Ros96a] *i.e.* as soon as we perform an *Apply* or *Copy*, we immediately discard any garbage. This ensures that we do not 'waste time' reducing garbage.

Garbage-free reduction also has a theoretical advantage. Rose notes that for $\lambda\mathrm{xgc}$, *"infinite reductions consist mainly of reductions inside garbage"* [Ros96a]. As we show in Section 2.2.4, this is true even more so for Λ_{sub} as $\overrightarrow{\mathrm{ACD}}$ allows copying between substitutions. This leads to yet more cases of infinite reductions than in $\lambda\mathrm{xgc}$. Garbage-free reduction removes garbage whenever it is created and so avoids these infinite reduction sequences. We follow Rose by proving PSN first for $\Lambda_{\mathrm{sub}\downarrow\mathrm{D}}$ and then reason about PSN for $\Lambda_{\mathrm{subC}^\flat}$ and Λ_{sub}.

Remark. Our definition of garbage-free reduction is slightly different to that of Rose. In $\lambda\mathrm{xgc}$, garbage-free reduction was defined by $\overrightarrow{\mathrm{bx}\downarrow\mathrm{gc}}^{\gg} = (\overrightarrow{\mathrm{bx}} \cdot \overrightarrow{\mathrm{gc}}^{\gg})$. We have previously mentioned that the reduction relation $\overrightarrow{\mathrm{x}}$ is somewhat matched by $\overrightarrow{\mathrm{CD}}$ – this would initially suggest that garbage-free reduction for Λ_{sub} should be defined as $\overrightarrow{\mathrm{ACD}} \cdot \overrightarrow{\mathrm{D}}^{\gg}$. If we use that definition then the remaining proofs in this section still hold but it turns out that we may instead use the smaller relation defined above. The reason is that in the following proofs, any

[3] There is an extension $\lambda\mathrm{xc}^-$ of $\lambda\mathrm{xgc}$ which has weak (*i.e.* conditional) composition of substitutions and retains PSN. See [BG99, Blo97] and Section 2.3.5 for details.

garbage-free reduction path will begin at a garbage-free term M. Since M contains no garbage, a reduction path $M \xrightarrow[\mathrm{ACD}]{} M' \xrightarrow[\mathrm{D}]{}\!\!\!\!\!\rightarrow N$ must begin with an $\xrightarrow[\mathrm{A}]{}$ or $\xrightarrow[\mathrm{C}]{}$ reduction. As N is guaranteed to be garbage-free, the same holds for N and so defining garbage-free reduction as $\xrightarrow[\mathrm{ACD}]{} \cdot \xrightarrow[\mathrm{D}]{}\!\!\!\!\!\rightarrow$ is redundant for our purposes – in the proofs of PSN, reduction paths start at pure terms.

We can also look at the issue from another angle. If M is garbage-free and $M \xrightarrow[\mathrm{bx\downarrow gc}]{}\!\!\!\!\!\rightarrow N$ such that $M \xrightarrow[\mathrm{x}]{} M' \xrightarrow[\mathrm{gc}]{}\!\!\!\!\!\rightarrow N$ then the real role of the $\xrightarrow[\mathrm{x}]{}$ reduction is to seek and replace free occurences of a variable with some body of a substitution *i.e.* as M is garbage-free, a reduction $M \xrightarrow[\mathrm{xvgc}]{} M'$ is not possible. Such reductions $M \xrightarrow[\mathrm{x}]{} M' \xrightarrow[\mathrm{gc}]{}\!\!\!\!\!\rightarrow N$ essentially worm down inside the term M until they hit paydirt with a free occurrence of some sought-after variable. Any garbage encountered along the way is quietly discarded. The $\xrightarrow[\mathrm{C}]{}$ relation performs this seek-and-replace job of $\xrightarrow[\mathrm{x}]{}$ from the outside (without the seeking). The $\xrightarrow[\mathrm{D}]{}$ relation then removes the garbage. This is our intuition as to why $\xrightarrow[\mathrm{AC\downarrow D}]{}$ suffices to replace $\xrightarrow[\mathrm{bx\downarrow gc}]{}\!\!\!\!\!\rightarrow$ in the following proofs.

We first prove confluence and then PSN for $\Lambda_{\mathrm{sub}\downarrow\mathrm{D}}$.

Lemmas 11.

1. For all Λ_{sub}-terms M,
$$\begin{array}{ccc} M & \xrightarrow{\mathrm{ACD}} & N \\ {\scriptstyle\mathrm{D}}\downarrow & & \downarrow{\scriptstyle\mathrm{D}} \\ \downarrow_{\mathrm{D}}(M) & \xrightarrow[\mathrm{AC\downarrow D}]{=} & \downarrow_{\mathrm{D}}(N) \end{array}$$

2. For garbage-free M,
$$\begin{array}{ccc} M & \xrightarrow{\mathrm{ACD}}\!\!\!\!\!\rightarrow & N \\ & {\scriptstyle\mathrm{AC\downarrow D}}\searrow & \downarrow{\scriptstyle\mathrm{D}} \\ & & \downarrow_{\mathrm{D}}(N) \end{array}$$

Proof.

1. When $M \xrightarrow[\mathrm{D}]{} N$, the proof follows from $\xrightarrow[\mathrm{D}]{}$UN. When $M \xrightarrow[\mathrm{AC}]{} N$ and the reduction occurs inside garbage then $\downarrow_{\mathrm{D}}(M) \equiv \downarrow_{\mathrm{D}}(N)$. Otherwise, for $M \xrightarrow[\mathrm{A}]{} N$ we have two cases depending on whether the reduction creates garbage or not. These are depicted below (using Lemma 2.5),
$$\begin{array}{ccc} (\lambda x.P)Q & \xrightarrow{\mathrm{A}} & P\langle x := Q\rangle \\ {\scriptstyle\mathrm{D}}\downarrow & & \downarrow{\scriptstyle\mathrm{D}} \\ (\lambda x.\downarrow_{\mathrm{D}}(P))\downarrow_{\mathrm{D}}(Q) & \xrightarrow{\mathrm{A}} & \downarrow_{\mathrm{D}}(P)\langle x :=\downarrow_{\mathrm{D}}(Q)\rangle \\ & & \xrightarrow[\mathrm{D}]{} \downarrow_{\mathrm{D}}(P) \end{array}$$
where the dotted reduction occurs when $x \notin \mathrm{fv}(\downarrow_{\mathrm{D}}(P))$. So we reach $\downarrow_{\mathrm{D}}(N)$ from $\downarrow_{\mathrm{D}}(M)$ with one $\xrightarrow[\mathrm{AC\downarrow D}]{}$ reduction.

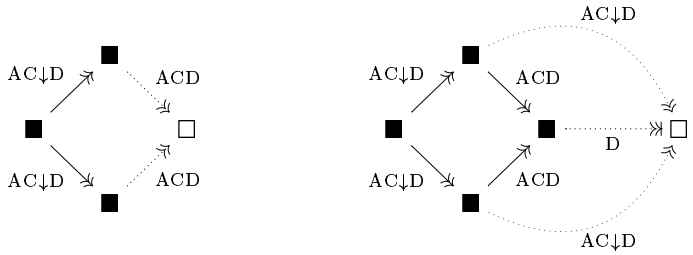
   A $M \xrightarrow[\mathrm{C}]{} N$ reduction outside garbage means that a free occurrence of some variable x is not discarded. A similar diagram sketches the proof.
$$\begin{array}{ccc} C[x]\langle x := Q\rangle & \xrightarrow{\mathrm{C}} & C[Q]\langle x := Q\rangle \\ {\scriptstyle\mathrm{D}}\downarrow & & \downarrow{\scriptstyle\mathrm{D}} \\ \downarrow_{\mathrm{D}}(C)[x]\langle x :=\downarrow_{\mathrm{D}}(Q)\rangle & \xrightarrow[\mathrm{C}]{} & \downarrow_{\mathrm{D}}(C)[\downarrow_{\mathrm{D}}(Q)]\langle x :=\downarrow_{\mathrm{D}}(Q)\rangle \\ & & \xrightarrow[\mathrm{D}]{} \downarrow_{\mathrm{D}}(C)[\downarrow_{\mathrm{D}}(Q)] \end{array}$$

2. This follows as
$$\begin{array}{ccccccccccc} M & \xrightarrow{\mathrm{ACD}} & M_1 & \xrightarrow{\mathrm{D}}\!\!\!\!\!\rightarrow & M_2 & \xrightarrow{\mathrm{ACD}} & M_3 & \xrightarrow{\mathrm{D}}\!\!\!\!\!\rightarrow & M_4 & \xrightarrow{\mathrm{ACD}} \cdots & \xrightarrow{\mathrm{D}}\!\!\!\!\!\rightarrow N \\ & {\scriptstyle\mathrm{AC\downarrow D}}\searrow & {\scriptstyle\mathrm{D}}\downarrow & & {\scriptstyle\mathrm{D}}\downarrow & & {\scriptstyle\mathrm{D}}\downarrow & & {\scriptstyle\mathrm{D}}\downarrow & & {\scriptstyle\mathrm{D}}\downarrow \\ & & \downarrow_{\mathrm{D}}(M_1) & = & \downarrow_{\mathrm{D}}(M_2) & \xrightarrow[\mathrm{AC\downarrow D}]{=} & \downarrow_{\mathrm{D}}(M_3) & = & \downarrow_{\mathrm{D}}(M_4) & \xrightarrow[\mathrm{AC\downarrow D}]{=} \cdots = & \downarrow_{\mathrm{D}}(N) \end{array}$$
clearly holds – as M is garbage-free, the left triangle is given by the definition of $\xrightarrow[\mathrm{AC\downarrow D}]{}$. The first square on the left is given by $\xrightarrow[\mathrm{D}]{}$UN, the next square is given by 1, and so on.

□

Theorem 12. $\xrightarrow[\mathrm{AC{\downarrow}D}]{}$ *CR.*

Proof. $\xrightarrow[\mathrm{AC{\downarrow}D}]{}$CR when $\xrightarrow[\mathrm{AC{\downarrow}D}]{}\!\!\!\!\!\!\rightarrow$ ◇ [Ter03, Proposition 1.1.10.iv]. The latter is shown as follows. The diagram on the left below can be filled in by noting that $\xrightarrow[\mathrm{AC{\downarrow}D}]{} \subseteq \xrightarrow[\mathrm{ACD}]{}\!\!\!\!\!\!\rightarrow$ and that $\xrightarrow[\mathrm{ACD}]{}$CR (which implies that $\xrightarrow[\mathrm{ACD}]{}\!\!\!\!\!\!\rightarrow$ ◇). The diagram on the right can then be filled in by applying Lemma 11.2 twice, noting that the terms at the starting points of the triangles are garbage-free.



□

Theorem 13 (PSN for $\Lambda_{\mathrm{sub}\downarrow\mathrm{D}}$). *Pure terms that are* $\xrightarrow[\beta]{}$*-strongly normalising are also strongly normalising for* $\xrightarrow[\mathrm{AC{\downarrow}D}]{}$.

Proof. Assume M is pure and strongly normalising for $\xrightarrow[\beta]{}$. Since M is pure it has no $\xrightarrow[\mathrm{CD}]{}$-redexes. Thus, every $\xrightarrow[\mathrm{AC{\downarrow}D}]{}\!\!\!\!\!\!\rightarrow$-reduction (finite or not) starting with M is of the form $M \equiv M_0 \xrightarrow[\mathrm{A}]{} M_1 \xrightarrow[\mathrm{CD}]{}\!\!\!\!\!\!\rightarrow M_2 \xrightarrow[\mathrm{A}]{} M_3 \xrightarrow[\mathrm{CD}]{}\!\!\!\!\!\!\rightarrow \cdots$ where the "$\xrightarrow[\mathrm{CD}]{}\!\!\!\!\!\!\rightarrow$" reductions are really of the form $\xrightarrow[\mathrm{D}]{}\!\!\!\!\!\!\rightarrow \cdot (\xrightarrow[\mathrm{C}]{} \cdot \xrightarrow[\mathrm{D}]{}\!\!\!\!\!\!\rightarrow) \cdots (\xrightarrow[\mathrm{C}]{} \cdot \xrightarrow[\mathrm{D}]{}\!\!\!\!\!\!\rightarrow)$ as we are working in $\xrightarrow[\mathrm{AC{\downarrow}D}]{}\!\!\!\!\!\!\rightarrow$.

Given any such reduction, we can construct the reduction graph as below:

$$\begin{array}{ccccccccccc} M \equiv & M_0 & \xrightarrow[\mathrm{A}]{} & M_1 & \xrightarrow[\mathrm{CD}]{}\!\!\!\!\!\!\rightarrow & M_2 & \xrightarrow[\mathrm{A}]{} & M_3 & \xrightarrow[\mathrm{CD}]{}\!\!\!\!\!\!\rightarrow & \cdots \\ & \Vert & & \downarrow_{\mathrm{CD}} & & \downarrow_{\mathrm{CD}} & & \downarrow_{\mathrm{CD}} & & \\ & \downarrow_{\mathrm{CD}}(M_0) & \xrightarrow[\beta]{+} & \downarrow_{\mathrm{CD}}(M_1) & \cdots & \downarrow_{\mathrm{CD}}(M_2) & \xrightarrow[\beta]{+} & \downarrow_{\mathrm{CD}}(M_3) & \cdots & \end{array}$$

where every second square starting from the leftmost square follows by Lemma 7.2 and where the other squares follow from $\xrightarrow[\mathrm{CD}]{}$UN.

M is strong normalising for $\xrightarrow[\beta]{}$ and so the lower reduction is finite. Since $\xrightarrow[\mathrm{CD}]{}$SN, the upper one must also be finite. □

2.2.2 PSN for $\Lambda_{\mathrm{subC}^\flat}$

Definition (inter-substitution reduction, $\xrightarrow[\mathrm{C}^\circ]{}$, $\xrightarrow[\mathrm{C}^\flat]{}$ reduction).

1. Inter-substitution reduction *is the contextual closure of the reduction generated by:*

$$C'\big[M\langle y := C[x]\rangle\big]\langle x := N\rangle \xrightarrow[\mathrm{C}^\circ]{} C'\big[M\langle y := C[N]\rangle\big]\langle x := N\rangle$$

2. $\xrightarrow[\mathrm{C}^\circ]{}$ *is the largest subrelation of* $\xrightarrow[\mathrm{C}]{}$ *which excludes any inter-substitution reduction.*

3. $\xrightarrow[\mathrm{C}^\flat]{}$ *is the largest subrelation of* $\xrightarrow[\mathrm{C}]{}$ *which excludes any inter-substitution reduction whose redex is not entirely located in a body of substitution.*

$\xrightarrow[\mathrm{C}^\circ]{}$ could also be described as excluding any $\xrightarrow[\mathrm{C}]{}$ reductions where the variable of the redex was located inside a substitution. $\xrightarrow[\mathrm{C}^\flat]{}$ could be described as excluding any $\xrightarrow[\mathrm{C}]{}$ reductions where the substitution definition is a top-level substitution and the variable of the redex lies inside another substitution. For example,

$$x\langle y := Q\langle z := C[w]\rangle\langle w := P\rangle\rangle \xrightarrow[\mathrm{ACD}]{} x\langle y := Q\langle z := C[P]\rangle\langle w := P\rangle\rangle$$

is not a $\overline{\rightarrow}_{\subset}$ reduction as the free occurence of w is located inside a substitution definition. It is a $\overline{\rightarrow}_{C^\flat}$ reduction as the $\overline{\rightarrow}_{C}$ redex is entirely contained inside a body of substitution $Q\langle z := C[w]\rangle\langle w := P\rangle$[4]. The reduction

$$Q\langle z := C[w]\rangle\langle w := P\rangle \ \overline{\rightarrow}_{\mathrm{ACD}} Q\langle z := C[P]\rangle\langle w := P\rangle$$

is again not a $\overline{\rightarrow}_{\subset}$ reduction. It is also not a $\overline{\rightarrow}_{C^\flat}$ reduction as the inter-substitution copy happens between two top-level substitutions. Clearly, $\overline{\rightarrow}_{\subset} \subset \overline{\rightarrow}_{C^\flat} \subset \overline{\rightarrow}_{C}$.

An inter-substitution reduction is a replacement of a free variable located inside a substitution definition with some term. This form of reduction is related to the notion of composition of substitutions (see Section 2.3) which has been known to break PSN in other calculi (see Sections 2.3.3, 2.3.4, and 2.3.6).

Definition ($\overline{\rightarrow}_{\mathrm{A}\subset\mathrm{D}}$, $\overline{\rightarrow}_{\mathrm{AC^\flat D}}$, $\Lambda_{\mathrm{sub}\subset}$, $\Lambda_{\mathrm{subC^\flat}}$). $\overline{\rightarrow}_{\mathrm{A}\subset\mathrm{D}} = (\overline{\rightarrow}_{\mathrm{AD}} \cup \overline{\rightarrow}_{\subset})$. $\overline{\rightarrow}_{\mathrm{AC^\flat D}} = (\overline{\rightarrow}_{\mathrm{AD}} \cup \overline{\rightarrow}_{C^\flat})$. *We denote their respective calculi as* $\Lambda_{\mathrm{sub}\subset}$ *and* $\Lambda_{\mathrm{subC^\flat}}$.

λxgc does not have a rule to compose substitutions. Therefore, it would be reasonable to hypothesize that an inductive proof of PSN for $\Lambda_{\mathrm{sub}\subset}$ would follow the inductive proof of PSN for λxgc. We strongly believe this but do not prove it here. Instead, we will use the inductive proof of PSN for λxgc to prove PSN for the slightly stronger $\Lambda_{\mathrm{subC^\flat}}$ calculus. However, the reasoning at each stage should also hold for $\Lambda_{\mathrm{sub}\subset}$. The reason we are able to reuse Rose's inductive proofs whilst allowing some inter-substitution reduction is that the proofs rest on a property subSN (see below) which states that bodies of substitutions are strongly normalising for $\overline{\rightarrow}_{\mathrm{ACD}}$. This property must be shown to be preserved by reduction on a subset of Λx which includes the strongly normalising pure terms. We will show that $\overline{\rightarrow}_{\mathrm{AC^\flat D}}$ does indeed preserve this property on a suitable subset. As an intuition, note that if $\mathrm{subSN}(M)$ then as $\overline{\rightarrow}_{\mathrm{AC^\flat D}}$ does not allow any inter-substitution reduction at top-level, any other inter-substitution reduction preserves subSN.

Lemma 14 ($\downarrow_{\mathrm{CD}}$=$\downarrow_{\mathrm{C^\flat D}}$=$\downarrow_{\subset\mathrm{D}}$).

Proof. We prove $\downarrow_{\mathrm{CD}}$=$\downarrow_{\subset\mathrm{D}}$which is sufficient. Given a term M, the innermost substitutions are subterms $P\langle x := Q\rangle$ of M such that P and Q are pure. Given such a subterm, reduce an innermost substitution:

$$M \equiv C[P\langle x := Q\rangle] \ \overline{\rightarrow}_{\subset}^{*} C[P\{x := Q\}\langle x := Q\rangle] \ \overline{\rightarrow}_{\mathrm{D}} C[P\{x := Q\}].$$

None of the $\overline{\rightarrow}_{\subset}$ reductions in the path are inter-substitution reductions. We may repeat this process until it ends (as $\overline{\rightarrow}_{\mathrm{CD}}$SN). The proof follows by $\overline{\rightarrow}_{\mathrm{CD}}$UN. □

Definition ($\mathrm{SN}_\beta, \mathrm{SN}_{\lambda\mathrm{xgc}}, \mathrm{SN}_{\Lambda_{\mathrm{sub}}}, \mathrm{SN}_{\Lambda_{\mathrm{subC^\flat}}}$). SN_β *is the set of strongly normalising* λ*-calculus terms.* $\mathrm{SN}_{\lambda\mathrm{xgc}}$, $\mathrm{SN}_{\Lambda_{\mathrm{sub}}}$, *and* $\mathrm{SN}_{\Lambda_{\mathrm{subC^\flat}}}$ *are the sets of* Λx *terms which are strongly normalising for* $\overline{\rightarrow}_{\mathrm{bxgc}}$, $\overline{\rightarrow}_{\mathrm{ACD}}$, *and* $\overline{\rightarrow}_{\mathrm{AC^\flat D}}$ *respectively.*

The next two definitions help us describe if the reduction paths of a term outside garbage (#gf) or inside substitutions (subSN) are finite. The intuition is that a term is strongly normalising for $\overline{\rightarrow}_{\mathrm{AC^\flat D}}$ if finite paths always exist both outside garbage and inside substitutions (Theorem 19).

Definition ($\#\mathrm{gf}(M)$). *For all terms* $M \in \Lambda x$*, define* $\#\mathrm{gf}(M)$ *to be the maximum length of garbage-free (* $\overline{\rightarrow}_{\mathrm{AC}\downarrow\mathrm{D}}$*) reduction paths starting in* $\downarrow_{\mathrm{D}}(M)$.

Definition ($\mathrm{subSN}_{\subset}(M), \mathrm{subSN}_{C^\flat}(M), \mathrm{subSN}(M)$). *The predicates* $\mathrm{subSN}_{\subset}(M)$, $\mathrm{subSN}_{C^\flat}(M)$, *and* $\mathrm{subSN}(M)$ *state that all bodies of substitutions in* M *are strongly normalising for* $\overline{\rightarrow}_{\mathrm{A}\subset\mathrm{D}}$-, $\overline{\rightarrow}_{\mathrm{AC^\flat D}}$-, *and* $\overline{\rightarrow}_{\mathrm{ACD}}$ *respectively.*[5].

We will not discuss $\mathrm{subSN}_{\subset}(M)$ or $\mathrm{subSN}_{C^\flat}(M)$ much here. We hypothesize that the former is sufficient to prove PSN for $\Lambda_{\mathrm{sub}\subset}$ but the latter is not sufficient to prove PSN for $\Lambda_{\mathrm{subC^\flat}}$ – the term $p\langle u := Z\rangle$ in Proposition 15 would be a counterexample to the main theorem of this section. For $\Lambda_{\mathrm{subC^\flat}}$, we need the stronger property subSN.

To demonstrate how #gf and subSN describe the finiteness of reduction sequences outside and inside garbage respectively, let $\Omega \equiv (\lambda v.vv)(\lambda w.ww)$; $\#\mathrm{gf}(\Omega) = \infty$ and $\mathrm{subSN}(\Omega)$ is true (there are no substitutions) whereas $\#\mathrm{gf}(x\langle y := \Omega\rangle) = 0$ and $\mathrm{subSN}(x\langle y := \Omega\rangle)$ is false, where $x \neq y$.

Notation (properties of terms and subsets of Λx). *We typically use the same notation for a property of a term and a subset of* Λx *e.g.* $\mathrm{subSN}(M)$ *means that* M *satisfies* subSN *whereas* $\#\mathrm{gf} < \infty$ *denotes the subset of* Λx *which satisfies this property.*

[4] Put another way, $\langle w := P\rangle$ is not a top-level substitution.

[5] In Bloo's terminology [Blo97], we would say that M is *decent*.

Definition ($\Lambda x^{<\infty}$). *We define a subset $\Lambda x^{<\infty}$ of Λx as:*

$$\Lambda x^{<\infty} = \{M \in \Lambda x \mid \forall N \subseteq M \cdot \downarrow_{\mathrm{CD}}(N) \in \mathrm{SN}_{\beta}\}^{6}$$

where $\subseteq$ denotes 'subterm (non-strict) of'.

The pure terms of $\Lambda x^{<\infty}$ are exactly the strongly normalising pure terms of the λ-calculus. As it excludes some non-terminating Λx terms, it seems a good starting point for the proof of PSN. In fact, $\Lambda x^{<\infty}$ characterises $\mathrm{SN}_{\lambda \mathrm{xgc}}$ [BR95] and we hypothesize that it also characterizes $\mathrm{SN}_{\Lambda_{\mathrm{subG}}}$. It does not characterize $\mathrm{SN}_{\Lambda_{\mathrm{sub}}}$ (see Section 2.2.4).

$\Lambda x^{<\infty}$ also does not characterize $\mathrm{SN}_{\Lambda_{\mathrm{subC^{\flat}}}}$. Define Z as:

$$Z \equiv z\langle x := yy\rangle\langle y := \lambda v.vv\rangle,$$

where all variables are distinct. The set of subterms of Z are (up to equivalence):

$$S = \{v, vv, \lambda v.vv, z\langle x := yy\rangle, Z\}.$$

Proposition 15. $\mathrm{SN}_{\Lambda_{\mathrm{subC^{\flat}}}} \neq \mathrm{SN}_{\lambda \mathrm{xgc}}$.

Proof. Consider the term $p\langle u := Z\rangle \in \Lambda x^{<\infty}$. By [BR95], $Z \in \mathrm{SN}_{\lambda \mathrm{xgc}}$. However,

$$Z \ \xrightarrow[\mathrm{C^{\flat}}]{} \ \xrightarrow[\mathrm{C^{\flat}}]{} p\langle u := z\langle x := \Omega\rangle\langle y := \lambda v.vv\rangle\rangle$$

and Proposition 6.2 shows that Z is not strongly normalising for $\xrightarrow[\mathrm{AC^{\flat}D}]{}$. □

Corollary 16. $\mathrm{SN}_{\Lambda_{\mathrm{subC^{\flat}}}} \subset \mathrm{SN}_{\lambda \mathrm{xgc}}$.

Proof. By definition, a Λx term which is not in $\Lambda x^{<\infty}$ has a subterm whose $\xrightarrow[\mathrm{CD}]{}$ normal form is not strongly normalising for β-reduction. Any infinite β-reduction sequence can be matched by an infinite $\xrightarrow[\mathrm{AC^{\flat}D}]{}$ sequence (Proposition 6.2 and Lemma 14). As $\mathrm{SN}_{\lambda \mathrm{xgc}} = \Lambda x^{<\infty}$, the set $\mathrm{SN}_{\Lambda_{\mathrm{subC^{\flat}}}}$ must be a subset of $\mathrm{SN}_{\lambda \mathrm{xgc}}$. Proposition 15 proves that it is a strict subset. □

We now define garbage-reduction which classifies all the 'useless' reductions which occur in garbage (and may lead to infinite sequences).

Definition (Garbage-reduction).

1. Garbage-reduction *is the contextual closure of the reduction generated by:*
   - *If $N \xrightarrow[\mathrm{ACD}]{} N'$ and $x \notin \mathrm{fv}(\downarrow_{\mathrm{D}}(M))$ then $M\langle x := N\rangle \xrightarrow[\mathrm{ACD}]{} M\langle x := N'\rangle$ is garbage reduction.*
   - *If $x \notin \mathrm{fv}(M)$ then $M\langle x := N\rangle \xrightarrow[\mathrm{ACD}]{} M$ is garbage-reduction.*
   - *If $x \notin \mathrm{fv}(\downarrow_{\mathrm{D}}(M))$ then*

     $$C[M\langle x := \ldots y \ldots\rangle]\langle y := N\rangle \xrightarrow[\mathrm{ACD}]{} C[M\langle x := \ldots N \ldots\rangle]\langle y := N\rangle$$

     is garbage reduction.
2. *Reduction* outside garbage *is any reduction that is not garbage-reduction.*

The first type of garbage-reduction reduces the term that will replace x but any free instance of x in M is contained inside garbage. The second type of garbage-reduction is simply garbage-collecting via $\xrightarrow[\mathrm{D}]{}$. The third type is special to Λ_{sub}. It describes the wide substitution of a variable within garbage. The context C in the definition is needed as the outer substitution may not be directly above the inner substitution. This garbage reduction is always a $\xrightarrow[\mathrm{C}]{}$ reduction and can lead to the infinite sequences in terms like $z\langle x := yy\rangle\langle y := \lambda v.vv\rangle$. In Λ_{subG}, this final type of garbage reduction does not occur. In $\Lambda_{\mathrm{subC^{\flat}}}$, these reductions are a special case of the first type as the redex must be located entirely in a body of substitution.

As any $\xrightarrow[\mathrm{D}]{}$ reduction is garbage-reduction, reduction outside garbage only pertains to the $\xrightarrow[\mathrm{A}]{}$ and $\xrightarrow[\mathrm{C}]{}$ reductions *e.g.* if $N \xrightarrow[\mathrm{AC}]{} N'$ and $x \in \mathrm{fv}(\downarrow_{\mathrm{D}}(M))$ then $M\langle x := N\rangle \xrightarrow[\mathrm{AC}]{} M\langle x := N'\rangle$ is outside garbage. Any $\xrightarrow[\mathrm{AC}]{}$ reduction whose redex is not totally or partially contained in a body of substitution is also outside garbage.

[6] Bloo denotes this set as $\lambda x^{<\infty}$ – we use $\Lambda x^{<\infty}$ to keep our notation consistent.

Propositions 17.

1. If $M \xrightarrow[\mathrm{ACD}]{} N$ is garbage-reduction then $\downarrow_{\mathrm{D}}(M) \equiv \downarrow_{\mathrm{D}}(N)$.
2. If $M \xrightarrow[\mathrm{ACD}]{} N$ is outside garbage then $\downarrow_{\mathrm{D}}(M) \xrightarrow[\mathrm{AC}\downarrow\mathrm{D}]{} \downarrow_{\mathrm{D}}(N)$. Furthermore,
   (a) if $M \xrightarrow[\mathrm{C}]{} N$ then $\downarrow_{\mathrm{D}}(M) \xrightarrow[\mathrm{C}]{} \xrightarrow[\mathrm{D}]{*} \downarrow_{\mathrm{D}}(N)$,
   (b) if $M \xrightarrow[\mathrm{C}^\flat]{} N$ then $\downarrow_{\mathrm{D}}(M) \xrightarrow[\mathrm{C}^\flat]{} \xrightarrow[\mathrm{D}]{*} \downarrow_{\mathrm{D}}(N)$.

Proof.

1. We split the proof on the three cases where garbage-reduction occurs:
   - Let $Q \xrightarrow[\mathrm{ACD}]{} Q'$ and $x \notin \mathrm{fv}(\downarrow_{\mathrm{D}}(P))$. Given a garbage-reduction
   $$P\langle x := Q\rangle \xrightarrow[\mathrm{ACD}]{} P\langle x := Q'\rangle,$$
   the proof follows by $\xrightarrow[\mathrm{D}]{}$UN and the diagram
   $$\begin{array}{ccc} P\langle x := Q\rangle & \xrightarrow{\mathrm{ACD}} & P\langle x := Q'\rangle \\ \mathrm{D}\big\downarrow & & \mathrm{D}\big\downarrow \\ \downarrow_{\mathrm{D}}(P)\langle x := Q\rangle & & \downarrow_{\mathrm{D}}(P)\langle x := Q'\rangle \\ \mathrm{D}\big\downarrow & & \mathrm{D}\big\downarrow \\ \downarrow_{\mathrm{D}}(P) & = & \downarrow_{\mathrm{D}}(P). \end{array}$$
   - Follows by $\xrightarrow[\mathrm{D}]{}$UN.
   - Let $x \notin \mathrm{fv}(\downarrow_{\mathrm{D}}(M))$. The proof follows by $\xrightarrow[\mathrm{D}]{}$UN and the diagram
   $$\begin{array}{ccc} C[M\langle x := \ldots y \ldots\rangle]\langle y := N\rangle & \xrightarrow{\mathrm{ACD}} & C[M\langle x := \ldots N \ldots\rangle]\langle y := N\rangle \\ \mathrm{D}\big\downarrow & & \mathrm{D}\big\downarrow \\ C[\downarrow_{\mathrm{D}}(M)\langle x := \ldots y \ldots\rangle]\langle y := N\rangle & & C[\downarrow_{\mathrm{D}}(M)\langle x := \ldots N \ldots\rangle]\langle y := N\rangle \\ \mathrm{D}\big\downarrow & & \mathrm{D}\big\downarrow \\ C[\downarrow_{\mathrm{D}}(M)]\langle y := N\rangle & = & C[\downarrow_{\mathrm{D}}(M)]\langle y := N\rangle. \end{array}$$
2. A reduction $M \xrightarrow[\mathrm{AC}]{} N$ outside garbage can be described as contracting a redex in M which exists in some garbage-free form in $\downarrow_{\mathrm{D}}(M)$. More precisely, the redex of a reduction outside garbage in a term M has a unique residual in the term $\downarrow_{\mathrm{D}}(M)$; there is at most one residual as no copying takes place and there is at least one residual by the definition of reduction outside garbage *i.e.* the residual cannot be discarded.

   The proof follows from the two diagrams in the proof of Lemma 11.1.
   (a) If $M \xrightarrow[\mathrm{C}]{} N$ then the free occurrence of the variable (say x) in the redex does not lie under substitution. In $\downarrow_{\mathrm{D}}(M)$, it still does not lie under substitution.
   (b) If $M \xrightarrow[\mathrm{C}^\flat]{} N$ then the entire redex is a subterm of a body of substitution. This remains true for the reduct of the redex in $\downarrow_{\mathrm{D}}(M)$.

□

The following proofs relate specifically to $\Lambda_{\mathrm{subC}^\flat}$. We hypothesize that they also hold for Λ_{subC}, replacing subSN with $\mathrm{subSN}_{\mathrm{C}}$.

Lemmas 18.

In $\Lambda_{\mathrm{subC}^\flat}$,

1. If $\mathrm{subSN}(M)$ and $M \xrightarrow[\mathrm{AC}^\flat\mathrm{D}]{} N$ is garbage-reduction, then $\mathrm{subSN}(N)$.
2. If $\mathrm{subSN}(M)$ then M is strongly normalising for garbage-reduction.

Proof.

1. This follows from the definition of garbage-reduction. For the first case, say $M \equiv P\langle x := Q\rangle \xrightarrow[\mathrm{ACD}]{} P\langle x := Q'\rangle \equiv N$. Since $\mathrm{subSN}(M)$, Q is strongly normalising for $\xrightarrow[\mathrm{ACD}]{}$-reduction. Then Q' is strongly normalising for $\xrightarrow[\mathrm{ACD}]{}$-reduction.

   For the second case, say $M \equiv P\langle x := Q\rangle \xrightarrow[\mathrm{ACD}]{} P \equiv N$. Then the bodies of substitution of N are a subset of those of M and so $\mathrm{subSN}(N)$.

   In $\Lambda_{\mathrm{subC}^\flat}$, the third case is a special case of the first[7]. The proof follows by induction over the structure of terms.

2. We begin by defining two interpretations for subSN-terms M. Let $h_1(M)$ be the maximum length of $\xrightarrow[\mathrm{ACD}]{}$-reduction paths inside bodies of substitutions of M. The value $h_1(M)$ is well defined as M has a finite number of substitutions and the body of each substitution is strongly normalising for $\xrightarrow[\mathrm{ACD}]{}$-reduction by $\mathrm{subSN}(M)$. Let $h_2(M)$ be the number of substitutions of top-level substitutions in M. Any garbage-reduction reduct of M will never have a greater number of top-level substitutions than M as a garbage-reduction of the form $\xrightarrow[\mathrm{A}]{}$ (which introduces new substitutions) will only occur under a substitution.

   Let $M \xrightarrow[\mathrm{ACD}]{} N$ be garbage reduction occuring under a body of substitution of M. Then $h_1(M) > h_1(N)$ as if $h_1(N) \geq h_1(M)$ then there exists a $\xrightarrow[\mathrm{ACD}]{}$-reduction path starting from M which is longer than $h_1(M)$, the maximum such path.

   Let $M \xrightarrow[\mathrm{ACD}]{} N$ be garbage reduction which does not occur under a body of substitution of M. By the definition of garbage reduction it must be that $M \xrightarrow[\mathrm{D}]{} N$ and the reduction throws away a top-level substitution. Hence, the number of top-level substitutions of M is reduced by 1 and $h_2(M) > h_2(N)$.

   For any garbage reduction $M \xrightarrow[\mathrm{ACD}]{} N$ either of $h_1(M)$ or $h_2(M)$ decreases while the other does not increase. Hence, garbage reduction is strongly normalising for subSN-terms.

□

Theorem 19. *If* $\#\mathrm{gf}(M) < \infty$ *and* $\mathrm{subSN}(M)$ *then* M *is strongly normalising for* $\xrightarrow[\mathrm{AC}^\flat\mathrm{D}]{}$*-reduction.*

Proof. We induct over $\#\mathrm{gf}(M)$.

Base case $\#\mathrm{gf}(M) = 0$. By Proposition 17.2, any reduction $M \xrightarrow[\mathrm{AC}^\flat\mathrm{D}]{} N$ must be garbage-reduction. Now, for any garbage reduction $M \xrightarrow[\mathrm{AC}^\flat\mathrm{D}]{} N$ we have by Proposition 17.1 that $\downarrow_{\mathrm{D}}(N) \equiv \downarrow_{\mathrm{D}}(M)$. Hence, $\#\mathrm{gf}(N) = 0$ and so by the same argument any reduction $N \xrightarrow[\mathrm{AC}^\flat\mathrm{D}]{} N'$ must also be garbage reduction. It follows that any reduction path starting at M contains only garbage reductions. As $\mathrm{subSN}(M)$, it follows by Lemma 18.2 that M is strongly normalising.

Induction hypothesis Suppose $\#\mathrm{gf}(M) > 0$. We assume that if $\#\mathrm{gf}(M') < \#\mathrm{gf}(M)$ and $\mathrm{subSN}(M')$ then M' is strongly normalising for $\xrightarrow[\mathrm{AC}^\flat\mathrm{D}]{}$. We call this induction hypothesis IH1.

Suppose there exists an infinite reduction path

$$M \equiv M_0 \xrightarrow[\mathrm{AC}^\flat\mathrm{D}]{} M_1 \xrightarrow[\mathrm{AC}^\flat\mathrm{D}]{} M_2 \xrightarrow[\mathrm{AC}^\flat\mathrm{D}]{} M_3 \xrightarrow[\mathrm{AC}^\flat\mathrm{D}]{} \cdots$$

We have assumed that $\mathrm{subSN}(M)$. By Lemma 18.2, M is then strongly normalising for garbage reduction and so there is m such that $M \xrightarrow[\mathrm{AC}^\flat\mathrm{D}]{}\!\!\!\!\!\!\rightarrow M_m$ is garbage-reduction and $M_m \xrightarrow[\mathrm{AC}^\flat\mathrm{D}]{} M_{m+1}$ is reduction outside garbage. By Propositions 17.1 and 17.2, we have

$$\#\mathrm{gf}(M_{m+1}) < \#\mathrm{gf}(M_m) = \#\mathrm{gf}(M) < \infty.$$

[7] This is not true for Λ_{sub} and this case breaks the lemma for that calculus.

$\mathrm{subSN}(M_m)$ by Lemma 18. If we can prove by induction on the structure of M_m that also $\mathrm{subSN}(M_{m+1})$ then we can invoke IH1 to show that M_{m+1} is strongly normalising for $\xrightarrow[\mathrm{AC^\flat D}]{}$-reduction. We call the new induction hypothesis IH2. We treat some cases below, noting that $M_m \xrightarrow[\mathrm{AC^\flat}]{} M_{m+1}$ by definition of reduction outside garbage.

Case $M_m \equiv (\lambda x.N)P \xrightarrow[\mathrm{AC^\flat D}]{} N\langle x := P\rangle \equiv M_{m+1}$. Bodies of substitutions in N and P are strongly normalising since they are also bodies of substitutions in $(\lambda x.N)P \equiv M_m$ and $\mathrm{subSN}(M_m)$. Also, $\#\mathrm{gf}(P) < \#\mathrm{gf}((\lambda x.N)P)$ and so by IH1 P is strongly normalising, thus $\mathrm{subSN}(N\langle x := P\rangle)$.

Case $M_m \equiv C[x]\langle x := P\rangle \xrightarrow[\mathrm{AC^\flat D}]{} C[P]\langle x := P\rangle \equiv M_{m+1}$. We are in $\Lambda_{\mathrm{subC^\flat}}$ so if this free occurence of x is located inside a body of substitution, then the definition $\langle x := P\rangle$ is located inside the same body and the next case addresses this situation.

Otherwise, x is located outside of a body of substitution[8]. x is replaced by P whose bodies of substitutions are strongly normalising as they are also bodies of substitutions in M_m. Otherwise M_m and M_{m+1} have identical bodies of substitution. Hence, $\mathrm{subSN}(C[P]\langle x := P\rangle)$.

Case $M_m \equiv N\langle x := P\rangle \xrightarrow[\mathrm{AC^\flat D}]{} N\langle x := P'\rangle \equiv M_{m+1}$ where $P \xrightarrow[\mathrm{AC^\flat D}]{} P'$. We know $\mathrm{subSN}(M_m)$ which implies $\mathrm{subSN}(N)$. P is strongly normalising for $\xrightarrow[\mathrm{ACD}]{}$ since $\mathrm{subSN}(M_m)$. Hence, P' is strongly normalising and $\mathrm{subSN}(N\langle x := P'\rangle)$.

Case $M_m \equiv NP \xrightarrow[\mathrm{AC^\flat D}]{} N'P \equiv M_{m+1}$ where $N \xrightarrow[\mathrm{AC^\flat D}]{} N'$. Then $\mathrm{subSN}(N')$ by IH2. As $\mathrm{subSN}(NP)$, we have $\mathrm{subSN}(P)$ and so $\mathrm{subSN}(N'P)$.

Case $M_m \equiv \lambda x.N \xrightarrow[\mathrm{AC^\flat D}]{} \lambda x.N' \equiv M_{m+1}$ where $N \xrightarrow[\mathrm{AC^\flat D}]{} N'$. Similar to the last case.

Thus, M_{m+1} is strongly normalising for $\xrightarrow[\mathrm{AC^\flat D}]{}$-reduction. This completes the proof as $M \xrightarrow[\mathrm{AC^\flat D}]{*} M_{m+1}$ is a finite sequence. □

Corollary 20 (PSN for $\Lambda_{\mathrm{subC^\flat}}$). $\xrightarrow[\mathrm{AC^\flat D}]{}$ *PSN of* $\xrightarrow[\beta]{}$.

Proof.

Case $\Rightarrow$. If M is pure then it has no substitutions and so $\mathrm{subSN}(M)$. If M is strongly normalising for $\xrightarrow[\beta]{}$-reduction then by Theorem 13, $\#\mathrm{gf}(M) < \infty$. We now apply Theorem 19.

Case $\Leftarrow$. By Proposition 6.2 and Lemma 14, infinite $\xrightarrow[\beta]{}$-reductions induce infinite $\xrightarrow[\mathrm{AC^\flat D}]{}$-reductions.

□

We should remark that there is no bigraphical equivalent for Λ_{subC} or $\Lambda_{\mathrm{subC^\flat}}$. There is no obvious way to disallow inter-substitution reduction without changing the nature of activity of controls in a bigraphical reactive system (and assuming a suitable notion of activity is definable).

2.2.3 The problem with inter-substitution reduction

The proof in the previous section does not hold for Λ_{sub}. This is because inter-substitution reduction in general can introduce cases of infinite sequences inside substitution which do not occur in $\lambda\mathrm{xgc}$ or $\Lambda_{\mathrm{subC^\flat}}$.

Proposition 21. $\mathrm{SN}_{\Lambda_{\mathrm{sub}}} \neq \mathrm{SN}_{\lambda\mathrm{xgc}}$.

Proof. Consider Z from the last section. $Z \in \Lambda\mathrm{x}^{<\infty}$ and so by [BR95], $Z \in \mathrm{SN}_{\lambda\mathrm{xgc}}$. However,

$$Z \xrightarrow[\mathrm{C}]{} \xrightarrow[\mathrm{C}]{} z\langle x := \Omega\rangle\langle y := \lambda v.vv\rangle \equiv Z'$$

and Proposition 6.2 shows that Z is not strongly normalising for $\xrightarrow[\mathrm{ACD}]{}$. □

[8] This case breaks this theorem for Λ_{sub} – consider the term $M\langle y := x(\lambda v.vv)\rangle\langle x := \lambda w.ww\rangle$ and the obvious $\xrightarrow[\mathrm{C}]{}$ reduction which does not preserve subSN.

Corollary 22. $\mathrm{SN}_{\Lambda_{\mathrm{sub}}} \subset \mathrm{SN}_{\Lambda_{\mathrm{subC^\flat}}} \subset \mathrm{SN}_{\lambda\mathrm{xgc}}$.

Proof. By definition, a $\Lambda\mathrm{x}$ term which is not in $\Lambda\mathrm{x}^{<\infty}$ has a subterm whose $\overrightarrow{\mathrm{CD}}$ normal form is not strongly normalising for β-reduction. Any infinite β-reduction sequence can be matched by an infinite $\overrightarrow{\mathrm{ACD}}$ sequence (Proposition 6.2). As $\mathrm{SN}_{\lambda\mathrm{xgc}} = \Lambda\mathrm{x}^{<\infty}$, the set $\mathrm{SN}_{\Lambda_{\mathrm{sub}}}$ must be a subset of $\mathrm{SN}_{\lambda\mathrm{xgc}}$. Proposition 21 proves that it is a strict subset. As Z is strongly normalising for $\overrightarrow{\mathrm{AC^\flat D}}$, a similar argument and Corollary 16 finish the proof. □

Bloo proposed that the crucial step in the inductive proof of PSN for $\lambda\mathrm{xgc}$ was that it was provable that given a term M in $\Lambda\mathrm{x}^{<\infty}$, if $\mathrm{subSN}(M)$ then all $\overrightarrow{\mathrm{bxgc}}$-reducts M' of M satisfied $\mathrm{subSN}(M')$. In the proposition above, $\mathrm{subSN}(Z)$ is true but $\mathrm{subSN}(Z')$ is false and the inductive proof fails for Λ_{sub}.

The reason it fails is that $\mathrm{subSN}(M)$ is not a strong enough predicate for Λ_{sub}. subSN is meant to capture the property that all reduction sequences that occur inside substitutions are strongly normalising. However, Λ_{sub} allows interactions between substitutions similar to composition of substitutions (see Section 2.3) which cannot occur in $\lambda\mathrm{xgc}$. Specifically, these interactions are inter-substitution reductions. Proposition 21 demonstrates how substitution may alter another body of a substitution such that subSN no longer holds. In explicit substitution calculi without composition of substitutions, this behaviour is not possible. For Λ_{sub}, we need a stronger property similar to subSN to provide for this behaviour.

As our property will consider a body of substitution and all substitutions above it, we first introduce some notation to make the remainder of the section more legible.

Notation (substitutions, superbody of substitution). *When considering a subterm N of some term M, the term*

$$N\langle y_1 := R_1\rangle\langle y_2 := R_2\rangle \dots \langle y_n := R_n\rangle$$

includes all the substitutions above N in M (where $\langle y_i := R_i\rangle$ lies below $\langle y_{i+1} := R_{i+1}\rangle$). This is abbreviated to $N\langle y_1 \dots y_n\rangle$. When N is a body of substitution, we say $N\langle y_1 \dots y_n\rangle$ is a superbody of substitution.

Definition (preSN**).** *The predicate $\mathrm{preSN}(M)$ states that all superbodies N of substitutions in M are strongly normalising for $\overrightarrow{\mathrm{ACD}}$.*

$\mathrm{preSN}(Z)$ does not hold as $(yy)\langle y := \lambda v.vv\rangle$ is not strongly normalising. The predicate seems strong but we do need to consider all inter-substitutions reduction. For example, the term

$$M \equiv ((N\langle x := yv\rangle P)\langle y := z\rangle Q)\langle z := \lambda w.ww\rangle\langle v := \lambda u.uu\rangle$$

does not satisfy $\mathrm{preSN}(M)$ and has an infinite reduction sequence inside substitutions.

Next, we redefine the proposed set of strongly normalising terms of Λ_{sub} to account for inter-substitution reduction.

Definition ($\Lambda\mathrm{x}_{\mathrm{sub}}{}^{<\infty}$**).** *We define a subset $\Lambda\mathrm{x}_{\mathrm{sub}}{}^{<\infty}$ of $\Lambda\mathrm{x}$ as:*

$$\Lambda\mathrm{x}_{\mathrm{sub}}{}^{<\infty} = \{M \in \Lambda\mathrm{x} \mid \forall N \subseteq M \cdot \downarrow_{\mathrm{CD}}(N\langle y_1 \dots y_n\rangle) \in \mathrm{SN}_\beta\}$$

where $\subseteq$ denotes 'subterm (non-strict) of'.

Any term $M \in \mathrm{SN}_{\Lambda_{\mathrm{sub}}}$ satisfies both $\mathrm{preSN}(M)$ and $M \in \Lambda\mathrm{x}_{\mathrm{sub}}{}^{<\infty}$. The pure terms in $\Lambda\mathrm{x}_{\mathrm{sub}}{}^{<\infty}$ are exactly those which are strongly normalising for $\overrightarrow{\beta}$. It is therefore a likely candidate for $\mathrm{SN}_{\Lambda_{\mathrm{sub}}}$ and the proof of PSN but unfortunately it contains terms which are not strongly normalising.

Examples 23 (terms which are not strongly normalising). *Here are some examples of terms which are not strongly normalising but satisfy at least one of $\#\mathrm{gf} < \infty$, preSN, or inclusion in $\Lambda\mathrm{x}_{\mathrm{sub}}{}^{<\infty}$. All variables are distinct.*

1. $z\langle x := yy\rangle\langle y := \lambda v.vv\rangle$
   This term only satifies $\#\mathrm{gf} < \infty$.
2. $N\langle x := yv\rangle\langle y := z\rangle\langle z := \lambda w.ww\rangle\langle v := \lambda u.uu\rangle$, $x \notin \mathrm{fv}(N)$
   This term only satifies $\#\mathrm{gf} < \infty$.

3. $(\lambda x.z\langle y := (\lambda v.vv)x\rangle)(\lambda u.uu), y \neq z$
This term satisfies all three properties. However, it reduces via a $\xrightarrow[\mathrm{A}]{}$ reduction to

$$z\langle y := (\lambda v.vv)x\rangle\langle x := \lambda u.uu\rangle$$

where preSN *does not hold.*

4. $(\lambda x.z\langle y := (\lambda v.vv)x\rangle)\langle w := p\rangle(\lambda u.uu), y \neq z$
This term satisfies all three properties. However, it reduces via a $\xrightarrow[\mathrm{D}]{}$ reduction (discarding $\langle w := p\rangle$) to 3 above.

5. $\big(\lambda w.(\lambda x.z\langle y := (\lambda v.vv)x\rangle)w\big)\big(\lambda u.uu\big), y \neq z$
This term satisfies all three properties. However, it reduces as

$$\begin{array}{ll} \xrightarrow[\mathrm{A}]{} & \big((\lambda x.z\langle y := (\lambda v.vv)x\rangle)w\big)\langle w := \lambda u.uu\rangle \\ \xrightarrow[\mathrm{C}]{}\xrightarrow[\mathrm{D}]{} & 3 \end{array}$$

The important terms to note above are 3-5. In term 3, a $\xrightarrow[\mathrm{A}]{}$ reduction breaks preSN by creating a new substitution above existing ones which introduces an infinite sequence. In term 4, a similar reduction occurs but first a $\xrightarrow[\mathrm{D}]{}$ reduction must unblock a $\xrightarrow[\mathrm{A}]{}$ redex. In term 5, a copy enables a $\xrightarrow[\mathrm{A}]{}$ redex to break preSN.

However, it is encouraging that our examples which satisfy subSN but may not terminate all arise from non-terminating pure terms. For example,

$$\begin{array}{rll} \big(\lambda y.(\lambda x.z)yy\big)\big(\lambda v.vv\big) & \xrightarrow[\mathrm{ACD}]{}{}^{*} & 1 \\ \Big(\lambda v.\Big(\lambda z.\big(\lambda y.(\lambda x.N)(yv)\big)z\Big)\Big(\lambda w.ww\Big)\Big)\Big(\lambda u.uu\Big) & \xrightarrow[\mathrm{ACD}]{}{}^{*} & 2 \\ \Big(\lambda x.\big(\lambda y.z\big)\big((\lambda v.vv)x\big)\Big)\Big(\lambda u.uu\Big) & \xrightarrow[\mathrm{ACD}]{}{}^{*} & 3 \\ \Big(\lambda w.\lambda x.\big(\lambda y.z\big)\big((\lambda v.vv)x\big)\Big)\big(p\big)\Big(\lambda u.uu\Big) & \xrightarrow[\mathrm{ACD}]{}{}^{*} & 4 \\ \Big(\lambda w.\lambda x.\big(\lambda y.z\big)\big((\lambda v.vv)x\big)w\Big)\Big(\lambda u.uu\Big) & \xrightarrow[\mathrm{ACD}]{}{}^{*} & 5 \end{array}$$

In their inductive proofs of PSN for λxgc, Bloo [Blo97] shows that subSN is preserved by reduction for terms in $\Lambda \mathrm{x}^{<\infty}$ while Rose [Ros96b] shows that subSN is preserved by reduction for terms where $\#\mathrm{gf} < \infty$. Unfortunately, the examples above demonstrate that even with the more restrictive $\Lambda \mathrm{x}_{\mathrm{sub}}{}^{<\infty}$, preSN is not preserved by reduction.

As noted above, the problem is that a sequence of $\xrightarrow[\mathrm{A}]{}$ reductions may introduce new substitutions above existing ones and this can break preSN. We could further constrain preSN with the following definition.

Definition ($\mathrm{bigSN}_{\mathrm{A}}$**).** *The predicate* $\mathrm{bigSN}_{\mathrm{A}}(M)$ *states that for all sequences* $M \xrightarrow[\mathrm{A}]{}{}^{*} M_i$, $\mathrm{preSN}(M_i)$.

However, $\xrightarrow[\mathrm{D}]{}$ reductions may unblock $\xrightarrow[\mathrm{A}]{}$ redexes as in Example 23.4 above. We would then require a stronger definition in order that preSN was preserved.

Definition ($\mathrm{bigSN}_{\mathrm{AD}}$**).** *The predicate* $\mathrm{bigSN}_{\mathrm{AD}}(M)$ *states that for all sequences* $M \xrightarrow[\mathrm{AD}]{}{}^{*} M_i$, $\mathrm{preSN}(M_i)$.

Clearly, $\mathrm{bigSN}_{\mathrm{AD}}$ is a necessary property for a term to be strongly normalising. However, now we have to prove that it is preserved under $\xrightarrow[\mathrm{ACD}]{}$ reduction. We started this investigation by weakening the condition to only consider reductions outside substitution.

Definition (Reduction inside substitution).

1. Reduction inside substitution *is the contextual closure of the reduction generated by:*
   - *If* $N \xrightarrow[\mathrm{ACD}]{} N'$ *then* $M\langle x := N\rangle \xrightarrow[\mathrm{ACD}]{} M\langle x := N'\rangle$ *is reduction inside substitution.*
   - $C[M\langle x := \ldots y \ldots\rangle]\langle y := N\rangle \xrightarrow[\mathrm{ACD}]{} C[M\langle x := \ldots N \ldots\rangle]\langle y := N\rangle$ *is reduction inside substitution.*
2. Reduction outside substitution *is any other reduction.*

We explored these definitions and were able to prove the following lemmas (the proofs are in Appendix A.1):

Lemma 24 ($\overrightarrow{\text{AD}}$ SN).

Lemmas 25 (preservation, reflection of preSN).

1. *If* $\text{preSN}(M)$ *and* $M \overrightarrow{\text{AD}} N$ *is inside substitution then* $\text{preSN}(N)$.
2. *If* $\text{preSN}(M)$ *and* $M \overrightarrow{\text{D}} N$ *is outside substitution then* $\text{preSN}(N)$.
3. *If* $\text{preSN}(M)$ *and* $M \overrightarrow{\text{C}} N$ *then* $\text{preSN}(N)$.
4. *If* $\text{preSN}(N)$ *and* $M \overrightarrow{\text{A}} N$ *is inside substitution then* $\text{preSN}(M)$.

These lemmas allowed us to redefine bigSN_{AD} as:

Definition (bigSN_{AD}). *The predicate* $\text{bigSN}_{\text{AD}}(M)$ *states that for all sequences* $M \overrightarrow{\text{AD}}^{\gg} M_i$ *containing only reductions outside substitution,* $\text{preSN}(M_i)$.

and then prove the following lemmas:

Lemmas 26 (preservation of bigSN).

1. *If* $\text{bigSN}_{\text{AD}}(M)$ *and* $M \overrightarrow{\text{AD}} M_1$ *then* $\text{bigSN}_{\text{AD}}(M_1)$.
2. *If* $\text{bigSN}_{\text{AD}}(M)$ *and* $M \overrightarrow{\text{C}} M_1$ *does not create any new* $\overrightarrow{\text{A}}$ *redexes outside substitution then* $\text{bigSN}_{\text{AD}}(M_1)$.

However, although this eases the proof/counterproof of $\text{bigSN}_{\text{AD}}(M)$ for an arbitrary term M, it is not true that either $\#\text{gf} < \infty \cap \text{bigSN}_{\text{AD}}$ or $\Lambda\text{x}_{\text{sub}}^{<\infty} \cap \text{bigSN}_{\text{AD}}$ is closed under reduction (Example 23.5 is a counterexample for both). This unfortunately[9] leads us to the following heavy-handed definition.

Definition ($\text{bigSN}_{\text{ACD}}$). *The predicate* $\text{bigSN}_{\text{ACD}}(M)$ *states that for all sequences* $M \overrightarrow{\text{ACD}}^{\gg} M_i$, $\text{preSN}(M_i)$.

It may be possible to weaken the definition to reductions outside substitution again but we do not attempt this. It is rather unsatisfactory when compared to Bloo and Rose's proofs for λxgc. Where they require a term to satisfy subSN, we require a much stronger property – the preservation of preSN through reduction. Another consequence of Example 23.5 is that $\text{SN}_{\Lambda_{\text{sub}}} \subset \Lambda\text{x}_{\text{sub}}^{<\infty}$ and we do not have a simple property to characterise $\text{SN}_{\Lambda_{\text{sub}}}$.

2.2.4 PSN for Λ_{sub}

Lemmas 27.

1. If $\text{preSN}(M)$ and $M \overrightarrow{\text{ACD}} N$ is garbage-reduction, then $\text{preSN}(N)$.
2. If $\text{preSN}(M)$ then M is strongly normalising for garbage-reduction.

Proof.

1. Proof by case split. Lemmas 25.1-25.3 cover all cases.
2. The proof proceeds as in Lemma 18.2. The only difference is that we define the measure $h_1(M)$ as the maximum length of garbage-reduction paths where the redexes are at least partially contained inside substitution. The value $h_1(M)$ is well-defined as M has a finite number of substitutions and, in particular, for each innermost body of substitution Q, $Q\langle y_1 \dots y_n \rangle$ is strongly normalising.

□

[9] We feel that bigSN_{AD} is a preferable definition to $\text{bigSN}_{\text{ACD}}$ as $\overrightarrow{\text{AD}}$ SN.

We mentioned previously that Lemmas 18.1 and 18.2 do not hold for Λ_{sub}. To illustrate this, consider the sequence:

$$\begin{array}{ll} & z\langle x := yy\rangle\langle y := \lambda v.vv\rangle \\ \xrightarrow[\mathrm{ACD}]{} & z\langle x := (\lambda w.ww)y\rangle\langle y := \lambda v.vv\rangle \\ \xrightarrow[\mathrm{ACD}]{} & z\langle x := \Omega\rangle\langle y := \lambda v.vv\rangle. \end{array}$$

This is a sequence of garbage-reductions in Λ_{sub} (but not in $\Lambda_{\mathrm{subC^\flat}}$). The second reduction disproves Lemma 18.1 in Λ_{sub} and the entire sequence disproves Lemma 18.2.

Theorem 28. *If* $\#\mathrm{gf}(M) < \infty$ *and* $\mathrm{bigSN}_{\mathrm{ACD}}(M)$ *then* M *is strongly normalising for* $\xrightarrow[\mathrm{ACD}]{}$*-reduction.*

Proof. We use induction on $\#\mathrm{gf}(M)$.

Base case $\#\mathrm{gf}(M) = 0$. By Proposition 17.2, any reduction $M \xrightarrow[\mathrm{ACD}]{} N$ must be garbage-reduction; if it was reduction outside garbage then there would be a contradictory garbage-free reduction $\downarrow_{\mathrm{D}}(M) \xrightarrow[\mathrm{AC}\downarrow_{\mathrm{D}}]{} \downarrow_{\mathrm{D}}(N)$. Now, for any garbage reduction $M \xrightarrow[\mathrm{ACD}]{} N$, $\downarrow_{\mathrm{D}}(N) \equiv \downarrow_{\mathrm{D}}(M)$ by Proposition 17.1. Hence, $\#\mathrm{gf}(N) = 0$. It follows that any reduction path starting at M contains only garbage reductions. As $\mathrm{preSN}(M)$, it follows by Lemma 27.2 that M is strongly normalising.

Induction hypothesis Suppose $\#\mathrm{gf}(M) > 0$. We assume that if $\#\mathrm{gf}(M') < \#\mathrm{gf}(M)$ and $\mathrm{bigSN}_{\mathrm{ACD}}(M')$ then M' is strongly normalising for $\xrightarrow[\mathrm{ACD}]{}$. We call this induction hypothesis IH1.

Suppose there exists an infinite reduction path

$$M \equiv M_0 \xrightarrow[\mathrm{ACD}]{} M_1 \xrightarrow[\mathrm{ACD}]{} M_2 \xrightarrow[\mathrm{ACD}]{} M_3 \xrightarrow[\mathrm{ACD}]{} \cdots$$

We have assumed that $\mathrm{preSN}(M)$. By Lemma 27.2, M is then strongly normalising for garbage reduction and so there is a finite sequence $M \xrightarrow[\mathrm{ACD}]{\gg} M_m$ of garbage-reductions and $M_m \xrightarrow[\mathrm{ACD}]{} M_{m+1}$ is reduction outside garbage. By Propositions 17.1 and 17.2, we have

$$\#\mathrm{gf}(M_{m+1}) < \#\mathrm{gf}(M_m) = \#\mathrm{gf}(M) < \infty.$$

$\mathrm{bigSN}_{\mathrm{ACD}}(M_{m+1})$ as $M \xrightarrow[\mathrm{ACD}]{\gg} M_{m+1}$. Thus, M_{m+1} is strongly normalising for $\xrightarrow[\mathrm{ACD}]{}$-reduction. This completes the proof as $M \xrightarrow[\mathrm{ACD}]{\gg} M_{m+1}$ is a finite sequence.

□

Corollary 29. $\mathrm{SN}_{\Lambda_{\mathrm{sub}}} = (\#\mathrm{gf} < \infty \cap \mathrm{bigSN}_{\mathrm{ACD}})$

Proof. We must prove $\mathrm{SN}_{\Lambda_{\mathrm{sub}}} \subseteq (\#\mathrm{gf} < \infty \cap \mathrm{bigSN}_{\mathrm{ACD}})$. The contrapositive

$$\#\mathrm{gf}(M) = \infty \vee \neg\mathrm{bigSN}_{\mathrm{ACD}}(M) \Rightarrow M \notin \mathrm{SN}_{\Lambda_{\mathrm{sub}}}$$

follows by $\xrightarrow[\mathrm{AC}\downarrow_{\mathrm{D}}]{} \subseteq \xrightarrow[\mathrm{ACD}]{\gg}$. □

The proof of Theorem 28 is simpler than the corresponding proofs for λxgc or $\Lambda_{\mathrm{subC^\flat}}$ as a consequence of the fact that $\mathrm{bigSN}_{\mathrm{ACD}}$ is a much stronger property than subSN. As expected however, proving that a pure term satisfies $\mathrm{bigSN}_{\mathrm{ACD}}$ is much more complicated. We will need the following lemma.

Lemma 30 (delayed $\xrightarrow[\mathrm{A}]{}$ reduction).

$$\begin{array}{ccccc} M & \xrightarrow[\mathrm{ACD}]{\gg} & C_1[(\lambda x.P)Q] & \xrightarrow[\mathrm{ACD}]{} & C_1[P\langle x := Q\rangle] \xrightarrow[\mathrm{ACD}]{\gg} C_2[P''\langle x := Q'\rangle] \\ & & & \searrow^{\mathrm{ACD}} & \uparrow\!\uparrow_{\mathrm{ACD}} \\ & & & & C_2[(\lambda x.P')Q'] \xrightarrow[\mathrm{ACD}]{} C_2[P'\langle x := Q'\rangle] \end{array}$$

Proof. The proof follows from the fact that all controls in $'\Lambda_{\mathrm{BIG}}$ are active and reduction in $'\Lambda_{\mathrm{BIG}}$ matches that of Λ_{sub} [Mil05b]. The key is that what happens in $(\lambda x.P)Q$ stays in $(\lambda x.P)Q$ – the only effect that firing the $\xrightarrow[\mathrm{A}]{}$ redex has is that it allows free occurrences of x in P to be substituted which may lead to further reductions which happen entirely inside P. □

In the reduction paths of the lemma; (1) the firing of the $\xrightarrow[\mathrm{A}]{}$ redex does not affect Q *i.e.* any reductions totally inside Q or involving the surrounding context can still fire and so $Q \xrightarrow[\mathrm{ACD}]{}{}^{*} Q'$ on both the top and bottom lines; (2) neither $(\lambda x.P)Q$ or $P\langle x := Q\rangle$ can affect the surrounding context and so C_2 is like C_1 except with changes in any copied reducts of $(\lambda x.P)Q$ or $P\langle x := Q\rangle$; (3) on the bottom line, P' evolves from P by a combination of internal reductions and reductions involving the context. The bottom line joins the top line with a further combination of internal reductions within P, reductions involving the context, substitutions of Q for x, and reductions in any copies of $(\lambda x.P)Q$ or $P\langle x := Q\rangle$.

Corollary 31 (regression of terms). *Given a reduct $M_i \equiv C'[P''\langle x := Q'\rangle]$ of some term M where i) an ancestor of this substitution does not exist in M and ii) $P''\langle x := Q'\rangle$ is not contained inside substitution in M_i, we may 'regress' the term to $M_j \equiv C'[(\lambda x.P')Q']$ where $M \xrightarrow[\mathrm{ACD}]{}{}^{*} M_j \xrightarrow[\mathrm{ACD}]{}{}^{*} M_i$.*

This corollary will be needed in the proof of PSN to pull a body of substitution outside substitution to apply the inductive hypothesis.

Corollary 32 (PSN for Λ_{sub}). $\xrightarrow[\mathrm{ACD}]{}$ *PSN of* $\xrightarrow[\beta]{}$.

Proof. **Case $\Rightarrow$.** If M is strongly normalising for $\xrightarrow[\beta]{}$-reduction then by Theorem 13, $\#\mathrm{gf}(M) < \infty$. If we can prove $\mathrm{bigSN}_{\mathrm{ACD}}(M)$ then we can apply Theorem 28 to complete the proof.

We take as our induction hypothesis:

$$\text{If } M \text{ is pure and } \mathrm{maxred}_\beta(M) = n \text{ then } \mathrm{bigSN}_{\mathrm{ACD}}(M).$$

Note that by the induction hypothesis, if M is pure and $\mathrm{maxred}_\beta(M) = n$ then M is strongly normalising (by Theorems 13 and 28).

If $\mathrm{maxred}_\beta(M) = 0$ then $\mathrm{bigSN}_{\mathrm{ACD}}(M)$ is trivially true. If $\mathrm{maxred}_\beta(M) = 1$ then any $\xrightarrow[\mathrm{ACD}]{}$ reduct of M will only contain one body of substitution P which is pure such that $\mathrm{maxred}_\beta(P) = 0$. Hence, $\mathrm{bigSN}_{\mathrm{ACD}}(M)$. We will prove the inductive case by course-of-value induction.

Assume $\mathrm{bigSN}_{\mathrm{ACD}}(M)$ is false. There is then a finite sequence $M \xrightarrow[\mathrm{ACD}]{}{}^{*} M_m$ (depicted below) such that preSN holds at every step and then a reduction $M_m \xrightarrow[\mathrm{ACD}]{} M_{m+1}$ such that $\mathrm{preSN}(M_{m+1})$ is false. By Lemmas 25.1, 25.2, and 25.3, this must be a $\xrightarrow[\mathrm{A}]{}$-reduction outside substitution.

$$\begin{array}{ccccccccc}
M & \xrightarrow[\mathrm{A}]{} & M_1 & \xrightarrow[\mathrm{ACD}]{} & \cdots & \xrightarrow[\mathrm{ACD}]{} & M_m & \xrightarrow[\mathrm{A}]{} & M_{m+1} \\
\Vert & & \downarrow{\scriptstyle \mathrm{CD}} & & \downarrow{\scriptstyle \mathrm{CD}} & & \downarrow{\scriptstyle \mathrm{CD}} & & \downarrow{\scriptstyle \mathrm{CD}} \\
\downarrow_{\mathrm{CD}}(M) & \xrightarrow[\beta]{} & \downarrow_{\mathrm{CD}}(M_1) & \xrightarrow[\beta]{}{}^{*} & \cdots & \xrightarrow[\beta]{}{}^{*} & \downarrow_{\mathrm{CD}}(M_m) & \xrightarrow[\beta]{} & \downarrow_{\mathrm{CD}}(M_{m+1})
\end{array}$$

Let $M_m \equiv C[(\lambda x_l.N)R_l] \xrightarrow[\mathrm{ACD}]{} C[N\langle x_l := R_l\rangle] \equiv M_{m+1}$.
Let $\langle x_{l+1} := R_{l+1}\rangle \cdots \langle x_n := R_n\rangle$ be the substitutions above $(\lambda x_l.N)R_l$. We have assumed that $\mathrm{preSN}(M_{m+1})$ is false. Therefore, there exists some body of substitution $R_i, 1 \leq i \leq n$ such that its superbody $R_i\langle x_{i+1} \cdots x_n\rangle$ is not strongly normalising. As $\mathrm{preSN}(M_m)$, $1 \leq i \leq l$ – the problematic body of substitution is R_l or some body of substitution below it.

Assume that R_l is not strongly normalising. $\#\mathrm{gf}(M_m) < \infty$ so $\#\mathrm{gf}(R_l) < \infty$ and therefore $\mathrm{bigSN}_{\mathrm{ACD}}(R_l)$ is false so

$$R_l \xrightarrow[\mathrm{ACD}]{}{}^{*} C_1[(\lambda y.P)Q] \xrightarrow[\mathrm{A}]{} C_1[P\langle y := Q\rangle] \text{ and so}$$

$$M_m \xrightarrow[\mathrm{ACD}]{}{}^{*} C\left[(\lambda x.N)C_1[(\lambda y.P)Q]\right] \xrightarrow[\mathrm{A}]{} C\left[(\lambda x.N)C_1[P\langle y := Q\rangle]\right]$$

such that $\mathrm{preSN}(C_1[(\lambda y.P)Q])$ is true and $\mathrm{preSN}(C_1[P\langle y := Q\rangle)$ is false. We could then treat this case. If the new body of substitution Q is again not strongly normalising, we repeat this process. Eventually, we must reach a body which is strongly normalising as the term has a finite structure. Therefore, without loss of generality, we assume that R_l is strongly normalising.

We will break the proof over the cases $i = l$ and $i < l$. A proof that $R_i\langle x_{i+1} \cdots x_n\rangle$ is strongly normalising contradicts the assumption that $\mathrm{preSN}(M_{m+1})$ is false, yielding a proof by contradiction.

Case $i = l$**.** As R_l is strongly normalising, if $R_l\langle x_{l+1} \cdots x_n\rangle$ is not strongly normalising, there must be some substitution occuring between R_l and $\langle x_{l+1} \cdots x_n\rangle$. Now, each body of a garbage substitution S in R_l is also a body of substitution in M_m. Hence, the superbody of S is strongly normalising in M_{m+1} by $\mathrm{preSN}(M_m)$. As these substitutions S can only interact between themselves, we may discard them and only consider the term $\downarrow_{\mathrm{D}}(R_l)\langle x_{l+1} \cdots x_n\rangle$ to not be strongly normalising. Similarly, any substitutions in R_p, $l+1 \leq p \leq n$ which are garbage in $\downarrow_{\mathrm{D}}(R_l)\langle x_{l+1} \cdots x_n\rangle$ may be discarded and we need only consider the term $\downarrow_{\mathrm{D}}(R_l\langle x_{l+1} \cdots x_n\rangle)$.

We may now spend the remaining substitutions (without losing free variables by Proposition 5) and consider $\downarrow_{\mathrm{C}}\downarrow_{\mathrm{D}}(R_l\langle x_{l+1} \cdots x_n\rangle)$. Any infinite path inside the garbage of $\downarrow_{\mathrm{C}}\downarrow_{\mathrm{D}}(R_l\langle x_{l+1} \cdots x_n\rangle)$ can be replicated outside of garbage as all bodies of substitutions have been copied. Therefore, we may again discard all garbage and consider the term $\downarrow_{\mathrm{D}}\downarrow_{\mathrm{C}}\downarrow_{\mathrm{D}}(R_l\langle x_{l+1} \cdots x_n\rangle) \equiv \downarrow_{\mathrm{CD}}(R_l\langle x_{l+1} \cdots x_n\rangle)$. This is a subterm of $\downarrow_{\mathrm{CD}}(M_m)$, is pure, and has a β-reduction path less than that of M. Therefore, by I.H., it is strongly normalising for $\overrightarrow{\mathrm{ACD}}$ which is a contradiction.

Case $i < l$**.** We have a term

$$R_i\langle x_{i+1} \cdots x_{l-1}\rangle\langle x_l := R_l\rangle\langle x_{l+1} \cdots y_n\rangle$$

which we assume is not strongly normalising. If there exists garbage $\langle y := P\rangle$ inside R_i whose superbody in R_i is $P\langle z_1 \cdots z_p\rangle$ such that

$$T \equiv P\langle z_1 \cdots z_p\rangle\langle x_{i+1} \cdots x_n\rangle$$

is not strongly normalising then we consider that case, taking the superbody of the topmost subsitution of T such that its superbody is not strongly normalising. We could repeat this process a finite number of times.

Therefore, w.l.o.g., we assume that garbage in R_i does not help the term reduce infinitely. As in the previous case, we therefore consider the term

$$\downarrow_{\mathrm{D}}(R_i\langle x_{i+1} \cdots x_{l-1}\rangle\langle x_l = R_l\rangle\langle x_{l+1} \cdots y_n\rangle)$$

to not be strongly normalising. Again, we can spend and discard all substitutions and consider the pure term

$$V \equiv \downarrow_{\mathrm{CD}}(R_i\langle x_{i+1} \cdots x_{l-1}\rangle\langle x_l = R_l\rangle\langle x_{l+1} \cdots y_n\rangle).$$

If $x_i \in \mathrm{fv}(\downarrow_{\mathrm{D}}(M_m))$ then V is a pure subterm of $\downarrow_{\mathrm{CD}}(M_m)$ and hence is strongly normalising for $\overrightarrow{\mathrm{ACD}}$. Otherwise, we need to pull R_i out of the garbage by repeated application of Corollary 31.

If R_i is a top-level substitution of M_m then we apply Corollary 31 once to pull R_i out of substitution in some term M_k such that $M \overrightarrow{\mathrm{A}} \overrightarrow{\mathrm{ACD}}^{\gg} M_k \overrightarrow{\mathrm{ACD}}^{\gg} M_m$. The term $(\lambda x_i.U)R_i$ now lies under the substitutions $\langle x_{i+1} \cdots x_n\rangle$ and $\downarrow_{\mathrm{CD}}R_i\langle x_{i+1} \cdots x_n\rangle \subset \downarrow_{\mathrm{CD}}(M_k)$. As $\mathrm{maxred}_\beta(M_k) < \mathrm{maxred}_\beta(M)$, $\downarrow_{\mathrm{CD}}R_i\langle x_{i+1} \cdots x_n\rangle$ is strongly normalising by I.H.

If R_i is more deeply nested inside substitution then we apply Corollary 31 once for each level of nesting and the proof follows similarly.

Case $\Leftarrow$**.** By Proposition 6.2, infinite $\overrightarrow{\beta}$-reductions induce infinite $\overrightarrow{\mathrm{ACD}}$-reductions.

□

The author also attempted a proof based on replacing a β-redex $(\lambda x.P)Q$ in a pure term M with wQ where w was a fresh variable. The resulting term M' then has a smaller maximum β-reduction path. This was to be used in conjunction with an inductive hypothesis over $\mathrm{maxred}_\beta(M)$. Although the proof was abandoned, the idea of *origin tracking* [BKdV00] would seem to be the correct formalism we were searching for.

2.3 PSN and composition of substitutions

Before we present our proof of PSN for Λ_{sub} by simulation, we will discuss how composing substitutions in explicit substitution calculi may affect the PSN property.

Consider the β-reduction path

$$\big(\lambda x.(\lambda y.xyv)(zw)\big)\,u \xrightarrow[\beta]{} (\lambda y.uyv)(zw) \xrightarrow[\beta]{} u(zw)v.$$

In λxgc and Λ_{sub}, we have a reduction path

$$\begin{array}{rl} & \big(\lambda x.(\lambda y.xyv)(zw)\big)\,u \\ \longrightarrow & \big((\lambda y.xyv)(zw)\big)\langle x := u\rangle \\ \longrightarrow & (xyv)\langle y := zw\rangle\langle x := u\rangle. \end{array}$$

In λxgc, the outermost substitution $\langle x := u\rangle$ cannot be applied while it is above $\langle y := zw\rangle$ *i.e.* first $\langle y := zw\rangle$ must be pushed inside the term xyv and either applied or garbage-collected. In other words, substitutions are *blocked* by substitutions below them. It would be nice to allow substitutions to interact in some way by either swapping them with a rule like

$$M\langle x := N\rangle\langle y := P\rangle \longrightarrow M\langle y := P\rangle\langle x := N\langle y := P\rangle\rangle \tag{2.1}$$

or by allowing them to compose like

$$M\langle x := N\rangle\langle y := P\rangle \longrightarrow M\langle x := N\langle y := P\rangle\rangle \qquad \text{if } y \notin \mathrm{fv}(M)^{10} \tag{2.2}$$

which seems efficient if $x \in \mathrm{fv}(\downarrow_{\mathrm{CD}}(M))$ and $y \in \mathrm{fv}(N)$. Substitution calculi with such rules are said to allow *composition of substitutions*. The rules above are what we call *explicit composition*. We define explicit composition to be composition via a reduction rule whose only purpose is to compose substitutions. Rule (2.1) above is obviously unsafe for PSN as it immediately allows infinite paths of explicit compositions. In this section, we will demonstrate (using Bloo's examples [Blo97]) how the explicit nature of seemingly safe rules like (2.2) may break PSN for certain calculi.

2.3.1 Weak/full composition

Bloo [Blo97] defines *full composition of substitutions* (FCS) to mean that (i) any two adjacent substitutions can be composed and (ii) that it is possible that the outermost substitution may be evaluated before the innermost. Similarly, Kesner and Lengrand [KL05] define FCS to mean that any substitution in a term may always be immediately applied. A reduction system with Rule (2.1) above has FCS. However, as this rule is unsafe for termination and PSN, most calculi with FCS do not use it.

Weak composition of substitutions (WCS) is defined as conditional composition – composition may occur but only if some condition is satisfied. A reduction system where Rule (2.2) above was the only rule for composition would have WCS.

λxgc does not have any rule for composing substitutions. As we have seen, substitutions block substitutions above them. Extensions to λxgc with FCS and WCS are discussed in the following sections. We also explain how Λ_{sub} has FCS but the composition is not explicit.

2.3.2 Breaking PSN

When $\lambda\sigma$ and subsequent explicit substitution calculi were first introduced, it was assumed that they satisfied PSN. Melliès' counterexample [Mel95] for $\lambda\sigma$ was surprising and demonstrated that this seemingly natural property need not hold. $\lambda\sigma$ allows substitutions to be explicitly and fully composed. This composition combined with the distributive rules for pushing substitutions inside terms is the essence of Melliès' counterexample. $\lambda\sigma$ allows parallel substitutions but Bloo notes that this parallelism is not what breaks PSN but rather:

> "*the essential property for losing PSN is the possibility of moving one substitution from outside a second substitution to the* inside *of the latter by means of a composition of substitutions.*"[Blo97]

[10] The condition is necessary: consider $(yx)\langle x := N\rangle\langle y := P\rangle \longrightarrow (yx)\langle x := N\langle y := P\rangle\rangle$.

He shows this by first considering a similar calculus $\lambda\mathrm{x}\|\mathrm{c}$ with parallel substitutions and full composition and demonstrating that PSN is broken. He then drops the parallel construct to define a calculus $\lambda\mathrm{xc}$ with weak composition. The counterexample for PSN in $\lambda\mathrm{xc}$ is similar to that of $\lambda\mathrm{x}\|\mathrm{c}$ implying that parallel substitutions is not the essential property for losing PSN.

We will revisit Bloo's counterexamples for $\lambda\mathrm{x}\|\mathrm{c}$ and $\lambda\mathrm{xc}$ below, pointing out how the distributive rules and explicit composition are crucial for breaking PSN. We informally discuss Melliès' counterexample in the same light and then return our attention to Λ_{sub} which has no distributive rules or explicit composition.

2.3.3 $\lambda\mathrm{x}\|\mathrm{c}$

$\lambda\mathrm{x}\|\mathrm{c}$ [BR95, Blo97] is an extension of λxgc with parallel substitutions and FCS. The terms are defined inductively as:

$$M ::= x \quad | \quad \lambda x.M \quad | \quad MM \quad | \quad M\langle x_1, \ldots, x_m := M_1, \ldots, M_m\rangle.$$

The final term structure is an explicit parallel substitution where $m > 0$ and the variable convention applies *e.g.* the bound variables $x_1, \ldots, x_m$ are assumed to be distinct. The substitution is interpreted as a simultaneous substitution of M_i for x_i, $1 \leq i \leq m$ in the term M. The abbreviation $M\langle \vec{x} := \vec{M}\rangle$ denotes $M\langle x_1, \ldots, x_m := M_1, \ldots, M_m\rangle$.

The reduction relation in $\lambda\mathrm{x}\|\mathrm{c}$ is the union of $\xrightarrow[\mathrm{b}]{}$, $\xrightarrow[\mathrm{x}\|]{}$, and $\xrightarrow[\|\mathrm{c}]{}$. $\xrightarrow[\mathrm{b}]{}$ is as in λxgc and $\xrightarrow[\mathrm{x}\|]{}$ is defined similarly to $\xrightarrow[\mathrm{x}]{}$. The new reduction is $\xrightarrow[\|\mathrm{c}]{}$, defined as the contextual closure of

$$M\langle \vec{x} := \vec{P}\rangle\langle \vec{y} := \vec{Q}\rangle \xrightarrow[\|\mathrm{c}]{} M\langle \vec{x}, \vec{y} := P_1\langle \vec{y} := \vec{Q}\rangle, \ldots, P_m\langle \vec{y} := \vec{Q}\rangle, \vec{Q}\rangle.$$

This rule adds FCS to λxgc.

We now aim to give an intuition as to why the simply typable term M below, strongly normalising for β-reduction, is not strongly normalising in $\lambda\mathrm{x}\|\mathrm{c}$. We refer the reader to [Blo97] for an actual proof. In this example, we forget the variable convention to concentrate on the pattern of reduction. The reader may consider all bound variables in the terms to be subscripted with unique numbers.

$$\begin{aligned}
& M \equiv \lambda u.\big(\lambda x.(\lambda x.u)u\big)\big((\lambda x.u)u\big) && \\
\xrightarrow[\mathrm{b}]{} \quad & \lambda u.\big((\lambda x.u)u\big)\langle x := (\lambda x.u)u\rangle && (1)\\
\xrightarrow[\mathrm{x}\|]{}{}^{*} \quad & \lambda u.(\lambda x.u\langle x := (\lambda x.u)u\rangle)(u\langle x := (\lambda x.u)u\rangle) && (2)\\
\xrightarrow[\mathrm{b}]{} \quad & \lambda u.u\langle x := (\lambda x.u)u\rangle\langle x := u\langle x := (\lambda x.u)u\rangle\rangle && (3)\\
\xrightarrow[\|\mathrm{c}]{} \quad & \lambda u.u\langle x, x' := \underline{\big((\lambda x.u)u\big)\langle x := u\langle x := (\lambda x.u)u\rangle\rangle}, u\langle x := (\lambda x.u)u\rangle\rangle && (4)
\end{aligned}$$

The reduction sequence can be described as follows. The substitution in (1) is pushed inside the term by the distributive rule $\xrightarrow[\mathrm{x}\|]{}$. The pushing of the substitution inside the application/ $\xrightarrow[\mathrm{b}]{}$-redex $(\lambda x.u)u$ duplicates the substitution (2). The $\xrightarrow[\mathrm{b}]{}$ redex then fires, leaving the both copies of the substitution side by side (3). Note that the outer copy contains the original substitution. The copies are then composed (4).

The underlined subterm in (4) has an infinite reduction path which follows a similar pattern – the substitution is pushed inside the $\xrightarrow[\mathrm{b}]{}$ redex – copying itself, the redex fires, the substitutions are repeatedly composed until one lies just outside a $\xrightarrow[\mathrm{b}]{}$ redex, and the process repeats.

It is important to note that the infinite path above is made possible by the interplay between the distributive rules (which duplicate substitutions), the $\xrightarrow[\mathrm{b}]{}$ rule which places a substitution beside its descendant, and the composition rule which places a substitution inside its descendant. This last feature seems essential for losing PSN in calculi with explicit composition [Blo97, p.60].

We note that these infinite paths do not involve any substitutions being performed or garbage collected. It is the distributive nature of the $\xrightarrow[\mathrm{x}\|]{}$ and $\xrightarrow[\mathrm{x}]{}$ rules which require substitutions to be needlessly copied which can be dangerous. By 'needlessly copied' we mean that more copies of a substitution $\langle x := N\rangle$ can be made than free occurrences of x exist or that copies of a substitution are made whether or not free occurrences of x exist below all copies. Wide substitution therefore seems an important concept as its use avoids this creation of needless copying and local bigraphs are an appropriate test-bed for such research. Another solution to this problem is to always have exactly one free occurrence of a variable below a substitution so that the substitution is never needlessly copied. This is the approach taken in λlxr (see Section 1.3) which uses this linearity to keep PSN and FCS.

2.3.4 λxc

We have seen above that composition of substitutions can lead to infinite paths involving the distributive rules and substitution creation. However, the parallelism of substitutions in λx$\|$c does not seem to be the important factor for losing PSN. Bloo and Rose [BR95, Blo97] made this intuition precise by introducing λxc, which can be viewed as λx$\|$c without parallelism.

λxc shares the same set of terms as λxgc. The reduction relation of λxc is the union of $\xrightarrow[\text{bxgc}]{}$ and $\xrightarrow[\text{c}]{}$, where $\xrightarrow[\text{c}]{}$ is defined as the contextual closure of

$$M\langle x := P\rangle\langle y := Q\rangle \xrightarrow[\text{c}]{} M\langle x := P\langle y := Q\rangle\rangle \quad \text{if } y \notin \text{fv}(M).$$

This rule adds WCS to λxgc and is in fact the rule (2.2) introduced at the beginning of this section. It seems an efficient rule for the case where $x \in \text{fv}(\downarrow_{\text{xgc}}(M)), y \in \text{fv}(P)$.

Bloo [Blo97] shows that PSN is broken in λxc in a similar fashion to λx$\|$c.

$$\begin{aligned}
& M \equiv \lambda u.\big(\lambda x.(\lambda x.u)u\big)\big((\lambda x.u)u\big) \\
\xrightarrow[\text{b}]{} \quad & \lambda u.\big((\lambda x.u)u\big)\langle x := (\lambda x.u)u\rangle && (1)\\
\xrightarrow[\text{x}\|]{}{}^{\!\!*} \quad & \lambda u.(\lambda x.u\langle x := (\lambda x.u)u\rangle)(u\langle x := (\lambda x.u)u\rangle) && (2)\\
\xrightarrow[\text{b}]{} \quad & \lambda u.u\langle x := (\lambda x.u)u\rangle\langle x := u\langle x := (\lambda x.u)u\rangle\rangle && (3)\\
\xrightarrow[\text{c}]{} \quad & \lambda u.u\langle x := \underline{\big((\lambda x.u)u\big)\langle x := u\langle x := (\lambda x.u)u\rangle\rangle}\rangle && (4)
\end{aligned}$$

The infinite reduction begins in the underlined subterm in a similar manner. Again, the interplay between $\xrightarrow[\text{xap}]{}$, $\xrightarrow[\text{xab}]{}$, $\xrightarrow[\text{b}]{}$, and the composition rule breaks PSN.

This counterexample leads one to believe that in order to have PSN, an explicit substitution calculus should not create subterms inside substitutions which cannot be created outside substitutions. Bloo notes this in his dissertation and this intuition was the essence of our proof of PSN for Λ_{sub}. Armed with this intuition, Bloo and Geuvers [BG99] further constrained the composition of λxc and were able to show that the new calculus λxc$^-$ satisfied PSN.

2.3.5 λxc$^-$

λxc$^-$ shares the same set of terms as λxgc. The reduction relation of λxc$^-$ is the union of $\xrightarrow[\text{bxgc}]{}$ and $\xrightarrow[\text{c}^-]{}$, where $\xrightarrow[\text{c}^-]{}$ is defined as the contextual closure of

$$M\langle x := P\rangle\langle y := Q\rangle \xrightarrow[\text{c}^-]{} M\langle x := P\langle y := Q\rangle\rangle \quad \text{if } x \in \text{fv}(\downarrow_{\text{x}}(M)), y \notin \text{fv}(M).$$

This calculus is confluent, preserves strong normalisation, and has WCS. See [BG99] for details.

2.3.6 $\lambda\sigma$

We will not detail Melliès' counterexample for $\lambda\sigma$ as we would have to introduce too much notation at this late stage. However, the counterexample may be described as follows, where all rules are in $\lambda\sigma$.

Applications of the $Beta$ rule (akin to the $\xrightarrow[\text{b}]{}$ rule) create explicit substitutions. The App and Abs rules distribute these substitutions inside the term as in the examples above. A $Beta$ rule then creates a new substitution above the original one. The rules $Clos$, Map, and Ass then compose these two substitutions. Reduction sequences of this form continue indefinitely.

Again, the same interplay of similar rules yields the counterexample.

2.3.7 Λ_{sub}

As Λ_{sub} satisfies PSN, it seems natural to investigate how it allows composition of substitutions. Although there is no explicit composition rule, it has FCS – a substitution can always be performed whenever a free occurrence of a variable lies beneath the substitution definition regardless of what lies between. Two adjacent substitutions may also be *implicitly* composed.

By *implicit*, we mean the following. In λxgc, we could read $M\langle x := N\rangle\langle y := P\rangle$ as 'replace x with N in M *then* y with P in the result' as the inner substitution must be applied or discarded before the outer substitution

can be performed. In Λ_{sub}, we can read the same term as 'replace x with N in M or y with P in M or N.' Composition is allowed but not via an explicit reduction rule – it is allowed as $\overline{\mathrm{C}}^{\rangle}$ is a wide or non-local rule.

In Λ_{sub}, we also have reduction sequences like:

$$\begin{array}{ll} & M\langle x := N\rangle\langle y := P\rangle \\ \overline{\mathrm{C}}^{\rangle\!\rangle} & M\{y/P\}\langle x := N\{y/P\}\rangle\langle y := P\rangle \\ \overline{\mathrm{D}}^{\rangle} & M\{y/P\}\langle x := N\{y/P\}\rangle \end{array}$$

where $M\{y/P\}$ means 'the Λx term M with all free occurences of y replaced by P'. This sequence demonstrates how Rule (2.1) can be mimicked via wide substitution whilst avoiding the obvious infinite reductions.

Wide substitution is a very useful feature as it avoids the copying of a substitution definition using a distributive rule like $\overline{\mathrm{xap}}^{\rangle}$. This removes "*the possibility of bringing a substitution into a descendant of itself*" [Blo97] which leads to the counterexamples above. In fact, in the examples above PSN was broken without substitutions ever being perfomed. This is impossible in Λ_{sub} as $\overline{\mathrm{AD}}^{\rangle}$ SN. This substitution 'at a distance' allows Λ_{sub} to have FCS and PSN.

2.3.8 Confluence, PSN, and FCS

Bloo's dissertation contains a table of explicit substitution calculi [Blo97, Table 1.1] which summarises the properties of explicit substitution calculi at the time. The four properties are: level of confluence (closed/open)[11], termination of the substitution calculus[12], PSN, and level of substitution composition (none/WCS/FCS).

Out of the calculi in the table, only five satisfy closed confluence, PSN, and WCS. They are λxc^{-} (above), λ_d [Kes96], λ_{dn} [Kes96], λ_e [Kes96, FKP96], and $\lambda\zeta$ [Muñ96]. None of the calculi have FCS and the challenge of finding a calculus with closed confluence, PSN, and full composition seemed to be unanswered until recently when Kesner and Lengrand introduced λlxr [KL05, KL][13]. We have shown here that Λ_{sub} also shares these properties.

λlxr and Λ_{sub} seem to represent a big step forward for the field of explicit substitution[14] but although they share many similarities, they are based on different methods of substitution – λlxr uses distributive rules on linear terms whereas Λ_{sub} uses wide substitution. We spend the next section re-proving PSN for Λ_{sub} by using properties of λlxr.

[11] Open confluence means that the reduction system is confluent on terms with or without metavariables (open terms). Closed confluence means that the reduction system is confluent on terms without metavariables. Section 2.1 proves closed confluence for Λ_{sub}.

[12] The substitution calculus of Λ_{sub} is $\overline{\mathrm{CD}}^{\rangle}$ which is terminating (Proposition 1.3).

[13] The former reference also contains a summary of calculi which have PSN (although not FCS), the methods of retaining PSN, and the limitations that these methods impose.

[14] For historical reasons, we note that they were created independently around the same period (the draft paper in which Λ_{sub} originated was circulated in 2004).

2.4 Proof of PSN by simulation

We now prove PSN for Λ_{sub} by introducing a translation from Λ_{sub} to $\lambda\mathrm{lxr}$ and then showing that a reduction step in Λ_{sub} corresponds to a non-empty reduction sequence in the translation. The translation will be the composition of an encoding of Λ_{sub} terms in $\lambda\mathrm{lxr}$ and a normal form of $\lambda\mathrm{lxr}$. Our initial idea for the encoding is archived in Section 3.2. It is simpler than the one presented below but is insufficient for a proof of PSN via simulation.

2.4.1 The encoding of Λ_{sub} terms in $\lambda\mathrm{lxr}$

We begin with a slight alteration of the encoding of pure terms into $\lambda\mathrm{lxr}$ terms (up to the congruences in Figure 1.3) given in [KL05]. We alter their encoding by indexing translations $\mathcal{A}(M)$ with a set X of names which must include the free variables of M and none of the bound variables. This indexing is inspired by Milner's encoding of Λ_{sub} into $'\Lambda\mathrm{BIG}$ and is needed for our simulation of Λ_{sub} in $\lambda\mathrm{lxr}$ as the latter calculus remembers free variables which are discarded (interface preservation) whereas the former does not.

Definition (Encoding of pure terms in $\lambda\mathrm{lxr}$).

$$
\begin{array}{lll}
\mathcal{A}(x)_{X \uplus x} & := W_X(x) & \\
\mathcal{A}(\lambda x.M)_X & := W_{X \setminus \mathrm{fv}(M)}(\lambda x.\mathcal{A}(M)) & \text{if } x \in \mathrm{fv}(M) \\
\mathcal{A}(\lambda x.M)_X & := W_{X \setminus \mathrm{fv}(M)}(\lambda x.W_x(\mathcal{A}(M)) & \text{if } x \notin \mathrm{fv}(M) \\
\mathcal{A}(MN)_X & := W_{X \setminus (\mathrm{fv}(M) \cup \mathrm{fv}(N))}\left(C_\Phi^{\Delta,\Pi}\left(R_\Delta^\Phi(\mathcal{A}(M))\ R_\Pi^\Phi(\mathcal{A}(N))\right)\right) & \\
 & \qquad \text{where } \Phi := \mathrm{fv}(M) \cap \mathrm{fv}(N) &
\end{array}
$$

When a translation is not tagged, X is assumed to be $\emptyset$.

The weakening in the second encoding of $\lambda x.M$ is used to enforce linearity. It also blocks any garbage arising from an encoding of an $\xrightarrow[\mathrm{A}]{}$-redex $(\lambda x.M)N$, $x \notin \mathrm{fv}(M)$ from propagating through the term. The encoding of an application also enforces the linearity constraint – any free names shared in an application are renamed to be distinct. Contractions then explicitly bind them to their original names.

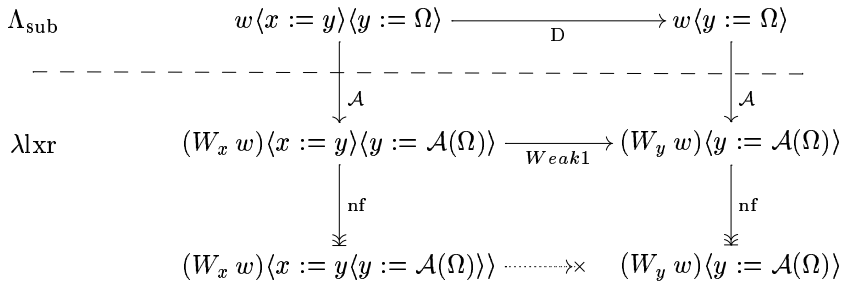
We now wish to extend this encoding to terms in Λ_{sub}. The remaining case is the term $M\langle x := N\rangle$. This term can arise from a $\xrightarrow[\mathrm{A}]{}$ reduction $(\lambda x.M)N \xrightarrow[\mathrm{A}]{} M\langle x := N\rangle$. We want our translation to be preserved by reduction in Λ_{sub} and so we first take the M' in $\mathcal{A}((\lambda x.M)N) \longrightarrow_B M'$ as our translation. We get the following (dropping the indices) when $x \in \mathrm{fv}(M)$:

$$
\begin{array}{rl}
 & \mathcal{A}((\lambda x.M)N) \\
= & C_\Phi^{\Delta,\Pi}\left(R_\Delta^\Phi(\mathcal{A}(\lambda x.M))\ R_\Pi^\Phi(\mathcal{A}(N))\right) \\
 & (\text{where } \Phi := \mathrm{fv}(\lambda x.M) \cap \mathrm{fv}(N)) \\
= & C_\Phi^{\Delta,\Pi}\left(R_\Delta^\Phi(\lambda x.\mathcal{A}(M))\ R_\Pi^\Phi(\mathcal{A}(N))\right) \text{ if } x \in \mathrm{fv}(M) \\
 & (x \notin \Phi) \\
= & C_\Phi^{\Delta,\Pi}\Big(\left(\lambda x.R_\Delta^\Phi(\mathcal{A}(M))\right)\ \left(R_\Pi^\Phi(\mathcal{A}(N))\right)\Big) \\
\longrightarrow_B & C_\Phi^{\Delta,\Pi}\Big(\ \left(R_\Delta^\Phi(\mathcal{A}(M))\right)\langle x := R_\Pi^\Phi(\mathcal{A}(N))\rangle\ \Big).
\end{array}
$$

The term inside the contractions reads as '$\mathcal{A}(M)\langle x := \mathcal{A}(N)\rangle$ where the shared free variables of $\mathcal{A}(M)$ and $\mathcal{A}(N)$ are renamed to be distinct'. The contractions then explicitly bind the renamed variables to their original names. This seems a sensible (linear) encoding of $M\langle x := N\rangle$.

However, a problem arises. In order to simulate wide substitution $\xrightarrow[\mathrm{C}]{}$ in $\lambda\mathrm{lxr}$ which has local substitution rules, we will need to employ a normal form in our translation which pushes all substitutions down to variables. We will explain why in more detail in Section 2.4.2 – for now, you could try to simulate the reduction $(x(xx))\langle x := y\rangle \xrightarrow[\mathrm{C}]{} (x(xy))\langle x := y\rangle$ in $\lambda\mathrm{lxr}$ using only the suggested encoding above. Now, if we let our translation be the composition of the encoding above composed with such a normal form then the reduction graph below can not be filled in (here we also assume that the translation of garbage $\langle y := \Omega\rangle$ introduces a weakening

W_y to enforce linearity).



The top square is easily filled in but the outer square can not be filled. This means that while the encoding should be sufficient to simulate $\xrightarrow[\mathrm{D}]{}$, the composition of the encoding and the normal form fails. The problem is that the outer subsitution $\langle y := \Omega\rangle$ is pushed inside the inner one during the translation and any discarding of the composed substitution in $\downarrow_{\mathrm{nf}} \circ \mathcal{A}$ loses both.

So, in order to simulate $\xrightarrow[\mathrm{ACD}]{}$ in $\lambda\mathrm{lxr}$, we need substitutions to be both pushed inside terms during the translation so that $\xrightarrow[\mathrm{C}]{}$ may be simulated, and also left outside so that they are not unfairly garbage-collected. This is the classic problem of trying to be in two places at once! To solve this dilemma, we create two different kinds of substitution in the encoding – one which may propagate through terms and one which cannot.

Definition (Encoding of Λ_{sub} terms in $\lambda\mathrm{lxr}$). *The encoding of Λ_{sub} terms in $\lambda\mathrm{lxr}$ is defined on pure terms as*

$$
\begin{array}{ll}
\mathcal{A}(x)_{X \uplus x} := W_X(x) & \\
\mathcal{A}(\lambda x.M)_X := W_{X\setminus \mathrm{fv}(M)}(\lambda x.\mathcal{A}(M)) & \text{if } x \in \mathrm{fv}(M) \\
\mathcal{A}(\lambda x.M)_X := W_{X\setminus \mathrm{fv}(M)}(\lambda x.W_x(\mathcal{A}(M)) & \text{if } x \notin \mathrm{fv}(M) \\
\mathcal{A}(MN)_X := W_{X\setminus(\mathrm{fv}(M)\cup\mathrm{fv}(N))}\left(C_\Phi^{\Delta,\Pi}\left(R_\Delta^\Phi(\mathcal{A}(M))\ R_\Pi^\Phi(\mathcal{A}(N))\right)\right) & \\
\qquad \text{where } \Phi := \mathrm{fv}(M) \cap \mathrm{fv}(N) &
\end{array}
$$

and on non-pure terms as

$$
\begin{array}{l}
\mathcal{A}(M\langle x := N\rangle)_X := W_Y\, C_\Phi^{\Pi,\Psi_1} C_{\Theta\setminus\Phi}^{\Gamma_2,\Psi_2}\left(W_{x'} C_\Pi^{\Delta,\Gamma_1}\left(\mathcal{A}(R_\Delta^\Phi(M))\langle x := N_1\rangle\right)\langle x' := N_2\rangle\right) \\
\text{where } x \in \mathrm{fv}(M), N_1 = \mathcal{A}(R_{\Gamma_2}^{\Theta\setminus\Phi} R_{\Gamma_1}^\Phi(N)), N_2 = \mathcal{A}(R_{\Psi_2}^{\Theta\setminus\Phi} R_{\Psi_1}^\Phi(N)) \\
\mathcal{A}(M\langle x := N\rangle)_X := W_Y\, C_\Phi^{\Delta,\Pi}\left(\ \left(W_x R_\Delta^\Phi(\mathcal{A}(M))\right)\langle x := R_\Pi^\Phi(\mathcal{A}(N))\rangle\ \right) \\
\text{where } x \notin \mathrm{fv}(M)
\end{array}
$$

where $\Phi = (\mathrm{fv}(M) \setminus \{x\}) \cap \mathrm{fv}(N)$, $\Theta = \mathrm{fv}(N)$, $Y = X \setminus (\mathrm{fv}(M) \cup \mathrm{fv}(N))$, and x' is a fresh name.

When a translation is not tagged, X is assumed to be $\emptyset$.

Example. *The $\{w, y, z\}$-indexed encoding of $(xxy)\langle x := wy\rangle$ is*

$$
W_z C_y^{yl_1,y3} C_w^{w_1,w_2}\left(W_{x'} C_{yl_1}^{y_1,y_2}\left((C_x^{x_1,x_2}(x_1x_2)y_1\right)\langle x := w_1y_2\rangle)\,\langle x' := w_2y_3\rangle\right).
$$

In the case where M and N have no common free variables ($x \notin \mathrm{fv}(N)$ by convention), the encoding of non-pure terms reads:

$$
\begin{array}{l}
\mathcal{A}(M\langle x := N\rangle)_X := W_Y C_\Theta^{\Gamma,\Psi}\left(W_{x'}\left(\mathcal{A}(M)\langle x := N_1\rangle\right)\langle x' := N_2\rangle\right) \\
\text{where } x \in \mathrm{fv}(M), N_1 = \mathcal{A}(R_\Gamma^\Theta(N)), N_2 = \mathcal{A}(R_\Psi^\Theta(N)) \\
\mathcal{A}(M\langle x := N\rangle)_X := W_Y\left(\ \left(W_x(\mathcal{A}(M))\right)\langle x := (\mathcal{A}(N))\rangle\ \right) \\
\text{where } x \notin \mathrm{fv}(M)
\end{array}
$$

where $\Theta = \mathrm{fv}(N)$, $Y = X \setminus (\mathrm{fv}(M) \cup \mathrm{fv}(N))$, and x' is a fresh name.

The encoding of pure terms makes sense – it forces linearity and nothing else. For non-pure terms, the encoding is more complicated. When translating a term $M\langle x := N\rangle$ where $x \in \mathrm{fv}(M)$, two substitutions are created in the $\lambda\mathrm{lxr}$ term, $\langle x := N_1\rangle$ and $\langle x' := N_2\rangle$. The second substitution with fresh name x' as binder is garbage and sits

$$
\begin{array}{rl}
 & (\lambda x.M)N \\
\longrightarrow_{Bs} & C^{\Gamma,\Psi}_{\Theta}\big(\big(W_{x'}(M\langle x := R^{\Theta}_{\Gamma}N\rangle)\big)\langle x' := R^{\Theta}_{\Psi}N\rangle\big) \\
\longrightarrow_{Weak1} & C^{\Gamma,\Psi}_{\Theta}\big(W_{\Psi}(M\langle x := R^{\Theta}_{\Gamma}N\rangle)\big) \\
\longrightarrow^{*}_{Merge} & R^{\Gamma}_{\Theta}(M\langle x := R^{\Theta}_{\Gamma}N\rangle) \\
\equiv & (M\langle x := N\rangle)
\end{array}
$$

Figure 2.1: Garbage collection of an idle substitution in $\lambda\mathrm{blxr}$

awaiting garbage collection. We call this the *idle* substitution as it can never propagate through the term. The first substitution may be pushed through the term to the free occurrences of x and so we call this the *mobile* substitution. Note that as $\mathcal{A}$ copies non-garbage substitutions, the encoding of terms with nested substitutions tends to get quite large. However, as we are only interested in proving a simulation, this is not of too much concern.

Idle and mobile substitutions will be used to simulate $\overline{\mathrm{ACD}}^{>}$ reduction. The translation will push the mobile substitution down into the term until a copy is at each free occurrence of the variable x (these occurrences will be uniquely named but 'mean x') whilst the idle substitution waits up top. In this way, mobile substitutions emulate the linking of a substitution to free variables as in $'\Lambda\mathrm{BIG}$ whereas the idle substitution sits at top level, emulating the actual structure of the $'\Lambda\mathrm{BIG}$ term. $\overline{\mathrm{C}}^{>}$ reductions are then mimicked by firing (Var) reductions which destroy a copy of the mobile substitution. When all these are performed, the idle substitution is garbage collected with a $(Weak1)$ reduction, mimicking a $\overline{\mathrm{D}}^{>}$ reduction.

We must also alter the reduction rules of $\lambda\mathrm{lxr}$ to create mobile and idle substitutions the moment a substitution is introduced *i.e.* in the $\longrightarrow_B$ rule. We define the following new rule to replace $\longrightarrow_B$.

Definition 33 ($\longrightarrow_{Bs}$). *The reduction $\longrightarrow_{Bs}$ is defined as the contextual closure, modulo $\equiv$, of the rule*

$$(\lambda x.M)N \longrightarrow_{Bs} C^{\Gamma,\Psi}_{\Theta}\big(\big(W_{x'}(M\langle x := R^{\Theta}_{\Gamma}N\rangle)\big)\langle x' := R^{\Theta}_{\Psi}N\rangle\big)$$

where $\Theta = \mathrm{fv}(N)$ and x' is a fresh name.

Definition ($\lambda\mathrm{blxr}$). *We let $\lambda\mathrm{blxr}$ denote the calculus obtained from $\lambda\mathrm{lxr}$ by replacing (B) with (Bs). We let $\longrightarrow_{\lambda\mathrm{blxr}}$ denote the reduction relation of $\lambda\mathrm{blxr}$. $\longrightarrow^{*}_{\lambda\mathrm{blxr}}$ denotes the reflexive and transitive closure of $\longrightarrow_{\lambda\mathrm{blxr}}$.*

Notation (creates garbage). *We that that a reduction $M \longrightarrow_{Bs} M'$* creates garbage *when the abstraction in the redex binds a weakening rather than an occurrence of a variable.*

Note that an idle substitution $\langle x' := N_2\rangle$ may always be garbage collected as in Figure 2.1 such that $\longrightarrow_B$ can be mimicked in $\lambda\mathrm{blxr}$ but we will not do so in our simulation unless no corresponding mobile substitutions $\langle x := N_1\rangle$ exist.

To aid our proof, we define labelled contexts for Λ_{sub} and $\lambda\mathrm{blxr}$. The next two definitions are included for formality and should not be necessary in order to read the proof.

Definition. *A* labelled Λ_{sub} context *$C[\ ]_X$ is a term with a hole in it where $\mathrm{fv}([\ ]_X) = X$. The term $C[M]_X$ is defined as long as $\mathrm{fv}(M) = X$ by filling the hole with M. Similarly, a* labelled $\lambda\mathrm{blxr}$ context *$C[\ ]_{\vec{X}}$ is a term with n holes which may be filled by a vector of n terms $\vec{P} = P_1, \cdots, P_n$ where $P_i = R^{\mathrm{fv}(M)}_{X_i}(P)$, $X_i \cap X_j = \emptyset$ for $1 \leq i \leq j \leq n$.*

We define $\lambda\mathrm{blxr}$ contexts in this way for our proof of PSN where each P_i will be a copy of an encoding of a Λ_{sub} redex, the copies being generated by encodings of non-garbage substitutions. We finally extend $\mathcal{A}$ to encoding contexts, in the appropriate manner.

Definition (extending $\mathcal{A}$ to contexts). *The encoding of Λ_{sub} contexts in $\lambda\mathrm{lxr}$ is defined as*

$$
\begin{array}{ll}
\mathcal{A}(\lambda x.[\ ]_Y)_X := W_{X\setminus Y}(\lambda x.[\ ]_Y) & \text{if } x \in \mathrm{fv}(M) \\
\mathcal{A}(\lambda x.[\ ]_Y)_X := W_{X\setminus Y}(\lambda x.W_x([\ ]_Y) & \text{if } x \notin \mathrm{fv}(M) \\
\mathcal{A}([\ ]_Y\ N)_X := W_{X\setminus(Y\cup\mathrm{fv}(N))}\big(C^{\Delta,\Pi}_{\Phi}\big(R^{\Phi}_{\Delta}([\ ]_Y)\ R^{\Phi}_{\Pi}(\mathcal{A}(N))\big)\big) & \\
\mathcal{A}(N\ [\ ]_Y)_X := W_{X\setminus(Y\cup\mathrm{fv}(N))}\big(C^{\Delta,\Pi}_{\Phi}\big(R^{\Phi}_{\Delta}(\mathcal{A}(N))\ R^{\Phi}_{\Pi}([\ ]_Y)\big)\big) & \\
\qquad\qquad \text{where } \Phi := Y \cap \mathrm{fv}(N) &
\end{array}
$$

and on non-pure terms as:

$$\mathcal{A}([\ ]_Z\langle x := N\rangle)_X := W_Y\, C_\Phi^{\Pi,\Psi_1} C_{\Theta\setminus\Phi}^{\Gamma_2,\Psi_2}\left(W_{x'} C_\Pi^{\Delta,\Gamma_1}\left(R_\Delta^\Phi([\ ]_Z)\langle x := N_1\rangle\right)\langle x' := N_2\rangle\right)$$
$$\text{where } x \in Z, N_1 = \mathcal{A}(R_{\Gamma_2}^{\Theta\setminus\Phi} R_{\Gamma_1}^\Phi(N)), N_2 = \mathcal{A}(R_{\Psi_2}^{\Theta\setminus\Phi} R_{\Psi_1}^\Phi(N))$$
$$\mathcal{A}([\ ]_Z\langle x := N\rangle)_X := W_Y\, C_\Phi^{\Delta,\Pi}\left(\ \left(W_x R_\Delta^\Phi([\ ]_Z)\right)\langle x := R_\Pi^\Phi(\mathcal{A}(N))\rangle\ \right)$$
$$\text{where } x \notin Z$$

where $\Phi = (Z \setminus \{x\}) \cap \mathrm{fv}(N)$, $\Theta = \mathrm{fv}(N)$, $Y = X \setminus (Z \cup \mathrm{fv}(N))$, *and* x' *is a fresh name and*

$$\mathcal{A}(M\langle x := [\ ]_Z\rangle)_X := W_Y\, C_\Phi^{\Pi,\Psi_1} C_{Z\setminus\Phi}^{\Gamma_2,\Psi_2}\left(W_{x'} C_\Pi^{\Delta,\Gamma_1}\left(\mathcal{A}(R_\Delta^\Phi(M))\langle x := [\ ]_{Z_1}\rangle\right)\langle x' := [\ ]_{Z_2}\rangle\right)$$
$$\text{where } x \in \mathrm{fv}(M), [\ ]_{Z_1} = R_{\Gamma_2}^{Z\setminus\Phi} R_{\Gamma_1}^\Phi([\ ]_Z), [\ ]_{Z_2} = R_{\Psi_2}^{Z\setminus\Phi} R_{\Psi_1}^\Phi([\ ]_Z)$$
$$\mathcal{A}(M\langle x := [\ ]_Z\rangle)_X := W_Y\, C_\Phi^{\Delta,\Pi}\left(\ \left(W_x R_\Delta^\Phi(\mathcal{A}(M))\right)\langle x := R_\Pi^\Phi([\ ]_Z)\rangle\ \right)$$
$$\text{where } x \notin \mathrm{fv}(M)$$

where $\Phi = (\mathrm{fv}(M) \setminus \{x\}) \cap Z$, $Y = X \setminus (\mathrm{fv}(M) \cup Z)$, *and* x' *is a fresh name. When a translation is not tagged,* X *is assumed to be* $\emptyset$.

Given a Λ_{sub} term $C[M]_X$, we have $\mathcal{A}(C[M]_X) = \mathcal{A}(C)[M_1, M_2, \ldots, M_n]_{\vec{X}}$ where $M_i = \mathcal{A}(R_{X_i}^{\mathrm{fv}(M)}(M))$ as the encoding may copy substitutions. In practice, we will only be interested in one copy of M and write $\mathcal{A}(C)[M_i]$, omitting the other copies and indexing.

Lemma 34 (properties of $\mathcal{A}$).

1. $\mathrm{fv}(M) = \mathrm{fv}(\mathcal{A}(M))$
2. $\mathrm{fv}(\mathcal{A}(M)_X) = X$
3. $\mathcal{A}(R_\Delta^\Phi(M)) = R_\Delta^\Phi(\mathcal{A}(M))$

From now on, we will adopt a convention for labelling the variables in contractions. The convention is explained in Appendix A.4 along with a graphical representation for the contraction structure of λlxr terms. Briefly, we index the variables which correspond to x in a linear term from the innermost to outermost, left to right and label the contractions accordingly.

We will also forget the indexing in the translation until Section 2.4.4. It will be required when considering the only rule of Λ_{sub} which loses free variables, $\xrightarrow[\mathrm{D}]{}$. The sections until then will focus on the problem of simulating wide substitution in λblxr and the indices will just confuse the discussion.

2.4.2 A normal form and the translation

We now reason whether the encoding $\mathcal{A}$ is a suitable translation for simulating $\xrightarrow[\mathrm{ACD}]{}$-reduction in λblxr. Consider

$$M \equiv (x(xx))\langle x := y\rangle \xrightarrow[\mathrm{C}]{} (x(xy))\langle x := y\rangle \equiv N.$$

We have $\mathcal{A}(M) = C_y^{yl_1,y_4}\left(W_{x'}\left(C_x^{x_1,xl_1}(x_1 C_{xl_1}^{x_2,x_3}(x_2x_3))\langle x := yl_1\rangle\right)\langle x' := y_4\rangle\right)$ and we wish to reduce this term to reach a term which is equivalent to $\mathcal{A}(N)$. As we must replace a free occurrence of x with y, we begin with the following reduction path:

$$\begin{array}{rl}
 & \mathcal{A}(M) \\
= & C_y^{yl_1,y_4}\left(W_{x'}\left(C_x^{x_1,xl_1}(x_1 C_{xl_1}^{x_2,x_3}(x_2x_3))\langle x := yl_1\rangle\right)\langle x' := y_4\rangle\right) \\
\xrightarrow[\lambda\mathrm{blxr}]{}{}^{*} & C_y^{yl_1,y_4}\left(W_{x'}\left(C_{yl_1}^{y_1,yl_2}(x_1\langle x_1 := y_1\rangle C_{yl_2}^{y_2,y_3}(x_2\langle x_2 := y_2\rangle x_3\langle x_3 := y_3\rangle))\right)\langle x' := y_4\rangle\right) \\
\xrightarrow[Var]{} & C_y^{yl_1,y_4}\left(W_{x'}\left(C_{yl_1}^{y_1,yl_2}(x_1\langle x_1 := y_1\rangle C_{yl_2}^{y_2,y_3}(x_2\langle x_2 := y_2\rangle y_3))\right)\langle x' := y_4\rangle\right) \\
\equiv & M'
\end{array}$$

However, $\mathcal{A}(N) \equiv C_y^{yl_1,y_4}\left(W_{x'}\left(C_{yl_1}^{y_3,yl_2}(C_x^{x_1,x_2}(x_1(x_2y_3))\langle x := yl_2\rangle)\right)\langle x' := y_4\rangle\right)$ and there is no reduction path $M' \longrightarrow_{\lambda\mathrm{blxr}} \mathcal{A}(N)$ as the mobile copies of $\langle x := y\rangle$ have been pushed inside the term in M' to allow the (Var) reduction to take place. This suggests that our translation should be $\mathcal{A}$ composed with a normal form.

We wish to define a normal form which pushes substitions in far as possible so that $\xrightarrow[\mathrm{C}]{}$ may be simulated in λlxr. If we try this approach with the example above, we get:

$$\begin{array}{rl} & \mathcal{A}(N) \\ = & C_y^{yl_1,y_4}\big(W_{x'}\big(C_{yl_1}^{y_3,yl_2}(C_x^{x_1,x_2}(x_1(x_2y_3))\langle x := yl_2\rangle)\big)\langle x' := y_4\rangle\big) \\ \xrightarrow[\lambda\mathrm{lxr}]{} & C_y^{yl_1,y_4}\big(W_{x'}\big(C_{yl_1}^{y_3,yl_2}(C_{yl_2}^{y_1,y_2}(x_1\langle x_1 := y_1\rangle(x_2\langle x_2 := y_2\rangle y_3)))\big)\langle x' := y_4\rangle\big) \\ \equiv & N' \end{array}$$

This almost matches M' except that the contractions are out of place. To fix this, we push the contractions inside as far as possible:

$$\begin{array}{rl} & N' \\ \equiv_{A,C1c} & C_y^{yl_1,y_4}\big(W_{x'}\big(C_{yl_1}^{y_1,yl_2}(C_{yl_2}^{y_2,y_3}(x_1\langle x_1 := y_1\rangle(x_2\langle x_2 := y_2\rangle y_3)))\big)\langle x' := y_4\rangle\big) \\ \longrightarrow_{CApp1} & C_y^{yl_1,y_4}\big(W_{x'}\big(C_{yl_1}^{y_1,yl_2}(x_1\langle x_1 := y_1\rangle C_{yl_2}^{y_2,y_3}(x_2\langle x_2 := y_2\rangle y_3))\big)\langle x' := y_4\rangle\big) \\ \equiv & M' \end{array}$$

This example leads us to the following definitions.

Definition (Normal form and the translation).

- $\longrightarrow_{Psh}$ *is defined to be the union of* (Abs), $(App1)$, $(App2)$, $(Weak2)$, $(Cont1)$, *and* $(Comp)$.
- $\longrightarrow_{Ctn}$ *is defined to be the union of* $(CAbs)$, $(CApp1)$, $(CApp2)$, $(CSubs)$, *and* $(Cross)$.
- $\longrightarrow_{\mathrm{p}\downarrow\mathrm{c}}(M)$ *is defined as the composition* $\longrightarrow_{Psh} \twoheadrightarrow_{Ctn}$.
- $\downarrow_{\mathrm{p}\downarrow\mathrm{c}}(M)$ *is defined as the normal form (up to* $\equiv$*) of* $\longrightarrow_{\mathrm{p}\downarrow\mathrm{c}}(M)$.
- *The translation* $\mathcal{T}$ *from* Λ_{sub} *to* λblxr *is defined on* Λ_{sub} *terms* M *as* $\mathcal{T}(M)_X \stackrel{\text{def}}{=} \downarrow_{\mathrm{p}\downarrow\mathrm{c}}(\mathcal{A}(M)_X)$.

The relation $\downarrow_{\mathrm{p}\downarrow\mathrm{c}}$ can be described as 'push one substitution inside and then push all contractions in as far as possible.' We prove it has a unique normal form in the next section.

Perhaps surprisingly (as it involves contractions), the $(Merge)$ reduction rule is omitted in the normal form. This is because the encoding only introduces weakenings bound by an explicit substitution or abstraction but not contractions. Therefore, the rule cannot be applied to a $\longrightarrow_{\mathrm{p}\downarrow\mathrm{c}}$-reduct of an encoding.

We now try the translation $\mathcal{T}$ on the problematic square at the beginning of this section to get:

$$\begin{array}{ccc} (xx)\langle x := y\rangle & \xrightarrow[\mathrm{C}]{} & (yx)\langle x := y\rangle \\ \big\downarrow \mathcal{T} & & \big\downarrow \mathcal{T} \\ \mathcal{T}((xx)\langle x := y\rangle) & \xrightarrow[Var]{} & \mathcal{T}((yx)\langle x := y\rangle) \end{array}$$

2.4.3 Contractions in the translation

All branches of the abstract syntax tree (AST) of a λlxr term occur at applications and substitutions. For any λlxr term M, $\downarrow_{\mathrm{p}\downarrow\mathrm{c}}(M)$ has all substitutions lying directly above either a weakening W_x or a free occurrence of x and all contractions $C_w^{y,z}$ pushed in as far as possible so that 1) at the first brancing in the AST below the contraction, the left branch of the application (resp. substitution) below the contraction contains a free occurrence of y or z and the right branch contains a free occurrence of the other or 2) both variables occur below the contraction and before any branch split.

Definition (contractions at their most efficient). *We say that the contractions in a* λlxr *term are* at their most efficient *when if* x_1 *and* x_2 *represent a variable* x*, there is a contraction* $C_{x_i}^{x_1,x_2}$ *just above (up to congruence) the split in the abstract syntax tree where* x_1 *is in one branch and* x_2 *in the other.*

For example, consider $C^{x_1,xl_1}_{x}\big(yC^{x_2,x_3}_{xl_1}((x_1x_2)x_3)\big)$ which represents $y((xx)x)$. This term does not have its contractions at their most efficient whereas $\mathcal{A}(y((xx)x)) \equiv y\big(C^{xl_1,x_3}_{x}(C^{x_1,x_2}_{xl_1}(x_1x_2))x_3\big)$ does. The former term can reduce to the latter however. It would appear that a $\longrightarrow_{Ctn}$-normal form of any $\lambda\mathrm{lxr}$ term has its contractions at their most efficient but we require a weaker statement to aid our proof of simulation.

Proposition 35. *Given a Λ_{sub} term M, any $\longrightarrow_{\mathrm{p}\downarrow\mathrm{c}}$-reduct M' of $\mathcal{A}(M)$ has its contractions at their most efficient.*

Proof. We induct over the length n of the $\longrightarrow_{\mathrm{p}\downarrow\mathrm{c}}$ path from $\mathcal{A}(M)$ to M' which is finite as $\longrightarrow_{\mathrm{p}\downarrow\mathrm{c}} \subset \longrightarrow_{\mathrm{xr}}$.

Base case: $\mathbf{n = 0}$. We must show that the encoding $\mathcal{A}(M)$ has the contractions at their most efficient. The proof follows by the definition of $\mathcal{A}$ and induction over the term structure (I.H.2).

The cases of abstractions and variables follow trivially.

In the encodings of $Q\langle x := P\rangle$ where $x \notin \mathrm{fv}(Q)$ and QP, the contractions of $\mathcal{A}(Q)$ and $\mathcal{A}(P)$ are at their most efficient by I.H.2. The contractions $C^{\Delta,\Pi}_{\Phi}$ in the encoding are then at their most efficient as $\Delta \subseteq \mathrm{fv}(\mathcal{A}(R^{\Phi}_{\Delta}(Q)))$ and $\Pi \subseteq \mathrm{fv}(\mathcal{A}(R^{\Phi}_{\Pi}(P)))$.

In the encoding of $Q\langle x := P\rangle$ where $x \in \mathrm{fv}(Q)$, the contractions of $\mathcal{A}(R^{\Phi}_{\Delta}(Q))$, $P_1 \equiv \mathcal{A}(R^{\Theta\setminus\Phi}_{\Gamma_2}R^{\Phi}_{\Gamma_1}(P))$, and $P_2 \equiv \mathcal{A}(R^{\Theta\setminus\Phi}_{\Psi_2}R^{\Phi}_{\Psi_1}(P))$ are at their most efficient by I.H.2. In the two outside contractions C^{Π,Ψ_1}_{Φ} and $C^{\Gamma_2,\Psi_2}_{\Theta\setminus\Phi}$, the set $\Pi \cup \Gamma_2$ binds names in the left branch whereas $\Psi_1 \cup \Psi_2$ binds names in the right branch. The inner contraction, binds the names of Δ in the left branch and Γ_1 in the right branch. Therefore, all contractions are at their most efficient.

Inductive case: $\mathbf{n = k + 1}$. Let $\mathcal{A}(M) \longrightarrow^{k}_{\mathrm{p}\downarrow\mathrm{c}} M'' \longrightarrow_{\mathrm{p}\downarrow\mathrm{c}} M'$. We assume that the contractions of M'' are at their most efficient and prove the same for M'. We only need consider the case where

$$M'' = C^{\Delta,\Pi}_{\Phi}(Q\langle x := P\rangle), \quad \Phi = (\mathrm{fv}(Q) \setminus \{x\}) \cap \mathrm{fv}(P), x \in \mathrm{fv}(Q)$$

and the substitution $\langle x := P\rangle$ is pushed inside the term Q. The proof follows by induction over the term structure. As contractions in M'' are at their most efficient, $\Delta \subseteq \mathrm{fv}(Q)$ and $\Pi \subseteq \mathrm{fv}(P)$. As $\langle x := Q\rangle$ is a mobile substitution, no weakening W_x lies beneath it. We break the proof over the possible $\longrightarrow_{Psh}$ reductions.

Case: (Abs), $Q \equiv \lambda y.Q'$

The contractions follow the substitution with $\longrightarrow_{CApp}$ reductions.

$$\begin{aligned} & C^{\Delta,\Pi}_{\Phi}((\lambda y.Q')\langle x := P\rangle) \\ \longrightarrow_{Psh} \quad & C^{\Delta,\Pi}_{\Phi}((\lambda y.Q'\langle x := P\rangle)) \\ \twoheadrightarrow_{Ctn} \quad & \lambda y.C^{\Delta,\Pi}_{\Phi}(Q'\langle x := P\rangle) \end{aligned}$$

The contractions of P and Q' remain at their most efficient.

Case: $(Weak2)$, $Q \equiv (W_y\, Q')$, $y \neq x$

As we are considering $\longrightarrow_{\mathrm{p}\downarrow\mathrm{c}}$-reducts of $\mathcal{A}(M)$, any weakening W_y is bound by an abstraction λy or an explicit substitution $\langle y := T\rangle$. Therefore, $y \notin \Delta \cup \Pi$. The contractions follow the substitution with $\longrightarrow_{Cross}$ reductions.

$$\begin{aligned} & C^{\Delta,\Pi}_{\Phi}((W_y\, Q')\langle x := P\rangle) \\ \longrightarrow_{Psh} \quad & C^{\Delta,\Pi}_{\Phi}(W_y(Q'\langle x := P\rangle)) \\ \twoheadrightarrow_{Ctn} \quad & W_y(C^{\Delta,\Pi}_{\Phi}(Q'\langle x := P\rangle)) \end{aligned}$$

The contractions of P and Q' remain at their most efficient.

Case: $(App1)$ and $(App2)$, $Q \equiv T_1T_2$

Let $\Delta = \Delta_1 \uplus \Delta_2$ such that $\Delta_1 \subseteq \mathrm{fv}(T_1)$ and $\Delta_2 \subseteq \mathrm{fv}(T_2)$. We then write $C^{\Delta,\Pi}_{\Phi}$ as $C^{\Delta_2,\Pi_2}_{\Phi_2}C^{\Delta_1,\Pi_1}_{\Phi_1}$. We treat the case where $x \in \mathrm{fv}(T_1)$. The contractions follow the substitution with $\longrightarrow_{CApp1}$ reductions.

$$\begin{aligned} & C^{\Delta_2,\Pi_2}_{\Phi_2}C^{\Delta_1,\Pi_1}_{\Phi_1}((T_1T_2)\langle x := P\rangle) \\ \longrightarrow_{Psh} \quad & C^{\Delta_2,\Pi_2}_{\Phi_2}C^{\Delta_1,\Pi_1}_{\Phi_1}(T_1\langle x := P\rangle T_2) \\ \twoheadrightarrow_{Ctn} \quad & C^{\Delta_2,\Pi_2}_{\Phi_2}\big((C^{\Delta_1,\Pi_1}_{\Phi_1}(T_1\langle x := P\rangle))T_2\big) \end{aligned}$$

The contractions of P, T_1, and T_2 remain at their most efficient.

Case: $(Comp)$, $Q \equiv T_1\langle y := T_2\rangle$

As above, let $\Delta = \Delta_1 \uplus \Delta_2$ such that $\Delta_1 \subseteq \mathrm{fv}(T_1)$ and $\Delta_2 \subseteq \mathrm{fv}(T_2)$ and we write the contractions as $C_{\Phi_2}^{\Delta_2,\Pi_2} C_{\Phi_1}^{\Delta_1,\Pi_1}$.

The contractions can follow with $\longrightarrow_{CSubs}$ reductions.

$$\begin{aligned} & C_{\Phi_1}^{\Delta_1,\Pi_1} C_{\Phi_2}^{\Delta_2,\Pi_2}(T_1\langle y := T_2\rangle\langle x := P\rangle) \\ \longrightarrow_{Psh} \quad & C_{\Phi_1}^{\Delta_1,\Pi_1} C_{\Phi_2}^{\Delta_2,\Pi_2}(T_1\langle y := T_2\langle x := P\rangle\rangle) \\ \twoheadrightarrow_{Ctn} \quad & C_{\Phi_1}^{\Delta_1,\Pi_1}(T_1\langle y := C_{\Phi_2}^{\Delta_2,\Pi_2}(T_2\langle x := P\rangle)\rangle) \end{aligned}$$

The contractions of P, T_1, and T_2 remain at their most efficient.

Case: $(Cont1)$, $Q \equiv C_x^{x_1,x_2}(Q')$

x is bound by the substitution and so $x \notin \Delta \cup \Pi$. Let $\Theta = \mathrm{fv}(P)$. As $\Pi \subseteq \mathrm{fv}(P) = \Theta$, we write the contractions created by the $\longrightarrow_{Cont}$ reduction as below.

$$\begin{aligned} & C_{\Phi}^{\Delta,\Pi}(C_x^{x_1,x_2}(Q')\langle x := P\rangle) \\ \longrightarrow_{Psh} \quad & C_{\Phi}^{\Delta,\Pi} C_{\Pi}^{\Pi_1,\Pi_2} C_{\Theta\setminus\Pi}^{\Theta_1,\Theta_2}(Q'\langle x_1 := P_1\rangle\langle x_2 := P_2\rangle) \end{aligned}$$

where $P_1 \equiv R_{\Theta_1}^{\Theta\setminus\Pi} R_{\Pi_1}^{\Pi}(P)$ and $P_2 \equiv R_{\Theta_2}^{\Theta\setminus\Pi} R_{\Pi_2}^{\Pi}(P)$. This last term is congruent to

$$C_{\Phi}^{\Pi_2,\Pi} C_{\Theta\setminus\Pi}^{\Theta_1,\Theta_2}(C_{\Pi}^{\Delta,\Pi_1}(Q'\langle x_1 := P_1\rangle)\langle x_2 := P_2\rangle)$$

and

$$C_{\Phi}^{\Pi_1,\Pi} C_{\Theta\setminus\Pi}^{\Theta_1,\Theta_2}(C_{\Pi}^{\Delta,\Pi_2}(Q'\langle x_2 := P_2\rangle)\langle x_1 := P_1\rangle).$$

□

Corollary 36. *For all Λ_{sub} terms M, the contractions of $\mathcal{T}(M)$ are at their most efficient.*

Corollary 37. *$\mathcal{A}$-images of Λ_{sub} terms have a unique $\longrightarrow_{\mathrm{p}\downarrow\mathrm{c}}$ normal forms (up to congruence).*

Lemma 38. *Let M be in $\longrightarrow_{\mathrm{p}\downarrow\mathrm{c}}$-normal form and let each weakening bound by an abstraction lie directly under (up to $\equiv$) that abstraction. If $M \longrightarrow_{\lambda\mathrm{blxr}} N$ is not a $\longrightarrow Bs$ reduction which does not create garbage then N is in $\longrightarrow_{\mathrm{p}\downarrow\mathrm{c}}$-normal form.*

Proof. We break the proof over the possible reductions, proving that the reductions do not create a $\longrightarrow_{Psh}$ redex which is sufficient.

Let $M \longrightarrow_{Bs} N$ create garbage. All substitutions in M are directly above (up to congruence) the weakening or the variable that they bind. This is true also of these substitutions in N. The reduction creates two substitutions. The idle one cannot move inside the term as it binds a weakening directly below it. As the reduction creates garbage, the mobile substitution binds a weakening which by assumption lies directly below it. Hence, it cannot move inside the term either.

The (Var) case is trivial.

In the $(WAbs)$, $(WApp1)$, $(WApp2)$, $(WSubs)$, and $(Merge)$ cases, the weakenings explicit in the redex and reactum are not bound by any substitution (as M is in $\longrightarrow_{\mathrm{p}\downarrow\mathrm{c}}$-normal form). In the $(Weak1)$ case, only the weakening explicit in the redex is bound by a substitution which is discarded through the reduction. Thus, the firing of these rules do not create any new $\longrightarrow_{Psh}$ redexes.

As M is in $\longrightarrow_{\mathrm{p}\downarrow\mathrm{c}}$-normal form, no substitutions bind the free variable of any contraction. Therefore, no $(CAbs)$, $(CApp1)$, $(CApp2)$, $(CSubs)$, or $(Cross)$ reduction creates any new $\longrightarrow_{Psh}$ redexes. □

Corollary 39. *Let $\mathcal{T}(M) \longrightarrow^{+}_{\lambda\mathrm{blxr}} N$ for some Λ_{sub} term M. Unless the reduction sequence contains a (Bs) reduction which does not create garbage or a $(CAbs)$ reduction, N is in $\longrightarrow_{\mathrm{p}\downarrow\mathrm{c}}$-normal form.*

Proof. It can be shown by induction that $\mathcal{A}(M)$ has any weakening bound by an abstraction lying directly under that abstraction. This is also true of $\mathcal{T}(M)$. The result follows by Lemma 38 noting that only a $(CAbs)$ reduction can come between a weakening and its binding abstraction (up to $\equiv$) as (Abs) redexes do not occur during the sequence. □

2.4.4 Proof of PSN by simulation

It seems reasonable to theorise that λblxr has the PSN property as it differs from λlxr only in the $\longrightarrow_{Bs}$ rule which creates two substitutions; one as normal and one which is 'garbage' and can not propagate through the term. This behaviour does not seem dangerous. It does not introduce new cases of infinite reductions as the garbage substitution may only interact with substitutions above it as the normal substitution can. Kesner and Lengrand [KL] prove that encodings of strongly normalising λ-terms in λlxr are strongly normalising using Lengrand's methods [Len05]. In other work, we have used those methods to prove the same for λblxr.

Theorem 40 (λblxr satisfies PSN [O'C06]). *For any pure term M, if $M \in \mathrm{SN}_\beta$ then $\mathcal{A}(M) \in \mathrm{SN}_{\lambda\mathrm{blxr}}$.*

Before we introduce our proof of simulation, we will give examples of the different cases, showing how a reduction step in Λ_{sub} may be matched by a non-empty sequence in λblxr. We omit the indexing in all cases but the $\xrightarrow[\mathrm{D}]{}$ case, where variables may be lost. In all the examples, the redex occurs inside a non-garbage body of substitution so that we may demonstrate how bodies of substitution are copied in the translation. We will also bend the variable convention for the purposes of demonstration.

For the $\xrightarrow[\mathrm{A}]{}$ case, there are two subcases depending on whether the new substitution is garbage or not.

Example 41 ($\xrightarrow[\mathrm{A}]{}$ simulation without garbage). Let

$$M \equiv x\langle x := (\lambda y.v(yy))z\rangle \xrightarrow[\mathrm{A}]{} x\langle x := (v(yy))\langle y := z\rangle\rangle \equiv N$$

with all variables distinct, $\Phi = \{v, z\}, \Phi_1 = \{v_1, z_1\}$, and $\Phi_2 = \{v_2, z_2\}$.

$$\begin{array}{rl}
 & \mathcal{T}(M) \\
\equiv & C_{vz}^{v_1 z_1, v_2 z_2}\big(W_{x'}(x\langle x := (\lambda y.v_1 C_y^{y_1 y_2}(y_1 y_2))z_1\rangle)\langle x' := (\lambda y.v_2 C_y^{y_1 y_2}(y_1 y_2))z_2\rangle\big) \\
\xrightarrow[Bs]{2} & C_{\Phi}^{\Phi_1,\Phi_2}\big(W_{x'}(x\langle x := C_{z_1}^{z_3 z_4}(W'_y((v_1 C_y^{y_1 y_2}(y_1 y_2))\langle y := z_3\rangle)\langle y' := z_4\rangle)\rangle) \\
 & \qquad\qquad \langle x' := C_{z_2}^{z_5 z_6}(W'_y((v_2 C_y^{y_1 y_2}(y_1 y_2))\langle y := z_5\rangle)\langle y' := z_6\rangle)\rangle)\big) \\
\equiv & \mathcal{A}(N) \\
\longrightarrow_{\mathrm{p}\downarrow\mathrm{c}}^{+} & C_{\Phi}^{\Phi_1,\Phi_2}\big(W_{x'}(x\langle x := C_{z_1}^{z_3 z_4}(W'_y(v_1 C_{z_3}^{z_7 z_8}(y_1\langle y_1 := z_7\rangle y_2\langle y_2 := z_8\rangle))\langle y' := z_4\rangle)\rangle) \\
 & \qquad\qquad \langle x' := C_{z_2}^{z_5 z_6}(W'_y(v_2 C_{z_5}^{z_9 z_0}(y_1\langle y_1 := z_9\rangle y_2\langle y_2 := z_0\rangle))\langle y' := z_6\rangle)\rangle)\big) \\
\equiv & \mathcal{T}(N)
\end{array}$$

The first line of reductions fires the $\longrightarrow_{Bs}$-redexes. The translation creates two copies of the $\xrightarrow[\mathrm{A}]{}$ redex so there are two corresponding $\longrightarrow_{Bs}$-reductions. In this example, we have now reached the term $\mathcal{A}(N)$. In general, we would now push the contractions created by the $\longrightarrow_{Bs}$-reductions to their most efficient points to reach a $\longrightarrow_{\mathrm{p}\downarrow\mathrm{c}}^{*}$-reduct of $\mathcal{A}(N)$ – the reduct where all substitutions except copies of the newly created one are pushed completely through the term. To match the term $\mathcal{T}(N)$, these new mobile substitutions are finally pushed inside the term.

Example 42 ($\xrightarrow[\mathrm{A}]{}$ simulation with garbage). Let

$$M \equiv x\langle x := (\lambda y.v(ww))z\rangle \xrightarrow[\mathrm{A}]{} x\langle x := (v(ww))\langle y := z\rangle\rangle \equiv N$$

with all variables distinct, $\Phi = \{w, v, z\}, \Phi_1 = \{w_1, v_1, z_1\}$, and $\Phi_2 = \{w_2, v_2, z_2\}$. This case is covered by the reduction sequence in Figure 2.1.

$$\begin{array}{rl}
 & \mathcal{T}(M) \\
\equiv & C_{\Phi}^{\Phi_1,\Phi_2}\big(W_{x'}(x\langle x := (\lambda y.W_y(v_1 C_{w_1}^{w_3 w_4}(w_3 w_4)))z_1\rangle) \\
 & \qquad\qquad \langle x' := (\lambda y.W_y(v_2 C_{w_2}^{w_5 w_6}(w_5 w_6)))z_2\rangle\big) \\
\xrightarrow[Bs]{2} & C_{\Phi}^{\Phi_1,\Phi_2}\big(W_{x'}(x\langle x := C_{z_1}^{z_3 z_4}(W_{y'}(W_y(v_1 C_{w_1}^{w_3 w_4}(w_3 w_4))\langle y := z_3\rangle)\langle y' := z_4\rangle)\rangle) \\
 & \qquad\qquad \langle x' := C_{z_2}^{z_5 z_6}(W_{y'}(W_y(v_2 C_{w_2}^{w_5 w_6}(w_5 w_6))\langle y := z_5\rangle)\langle y' := z_6\rangle)\rangle)\big) \\
\xrightarrow[Weak1]{2} & C_{\Phi}^{\Phi_1,\Phi_2}\big(W_{x'}(x\langle x := C_{z_1}^{z_3 z_4}(W_{z_4}(W_y(v_1 C_{w_1}^{w_3 w_4}(w_3 w_4))\langle y := z_3\rangle))\rangle) \\
 & \qquad\qquad \langle x' := C_{z_2}^{z_5 z_6}(W_{z_6}(W_y(v_2 C_{w_2}^{w_5 w_6}(w_5 w_6))\langle y := z_5\rangle))\rangle)\big) \\
\xrightarrow[Merge]{2\times 1} & C_{\Phi}^{\Phi_1,\Phi_2}\big(W_{x'}(x\langle x := W_y(v_1 C_{w_1}^{w_3 w_4}(w_3 w_4))\langle y := z_1\rangle\rangle) \\
 & \qquad\qquad \langle x' := W_y(v_2 C_{w_2}^{w_5 w_6}(w_5 w_6))\langle y := z_2\rangle\rangle\big) \\
\equiv & \mathcal{T}(N)
\end{array}$$

The first line of reductions fire the $\longrightarrow_{Bs}$-redexes. The translation creates two copies of the $\xrightarrow[\mathrm{A}]{}$ redex so there are two corresponding $\longrightarrow_{Bs}$-reductions. The encoding of garbage only creates one substitution so we now immediately garbage collect all the newly-created 'idle' substitutions. To reach $\mathcal{T}(N)$, we merge the resulting weakenings with contractions.

Observe that each term in this sequence is in $\longrightarrow_{\mathrm{p}\downarrow\mathrm{c}}$-normal form (as expected from Corollary 39).

Example 43 ($\xrightarrow[\mathrm{C}]{}$ simulation). We will pick up where Example 41 ended. Let

$$N \equiv x\langle x := (v(yy))\langle y := z\rangle\rangle \xrightarrow[\mathrm{C}]{} x\langle x := (v(zy))\langle y := z\rangle\rangle \equiv P$$

with all variables distinct, $\Phi = \{v, z\}$, $\Phi_1 = \{v_1, z_1\}$, and $\Phi_2 = \{v_2, z_2\}$.

$$\begin{array}{rl}
 & \mathcal{T}(N) \\
\equiv & C_{\Phi}^{\Phi_1,\Phi_2}\big(W_{x'}(x\langle x := C_{z_1}^{z_3 z_4}(W'_y(v_1 C_{z_3}^{z_7 z_8}(y_1\langle y_1 := z_7\rangle y_2\langle y_2 := z_8\rangle))\langle y' := z_4\rangle)) \\
 & \qquad \langle x' := C_{z_2}^{z_5 z_6}(W'_y(v_2 C_{z_5}^{z_9 z_0}(y_1\langle y_1 := z_9\rangle y_2\langle y_2 := z_0\rangle))\langle y' := z_6\rangle))\big) \\
\xrightarrow[Var]{2} & C_{\Phi}^{\Phi_1,\Phi_2}\big(W_{x'}(x\langle x := C_{z_1}^{z_3 z_4}(W'_y(v_1 C_{z_3}^{z_7 z_8}(z_7 y_2\langle y_2 := z_8\rangle))\langle y' := z_4\rangle)) \\
 & \qquad \langle x' := C_{z_2}^{z_5 z_6}(W'_y(v_2 C_{z_5}^{z_9 z_0}(z_9 y_2\langle y_2 := z_0\rangle))\langle y' := z_6\rangle))\big) \\
\equiv & \mathcal{T}(P)
\end{array}$$

This case is easier to explain. In a translated term, the substitutions are pushed in as far as possible. The variable being replaced has a substitution directly above it in the translation (for each copy of the variable). The sequence consists of two $\longrightarrow_{Var}$ reductions, one for each copy of the variable due to the translation. By Corollary 39, each term in this sequence is in $\longrightarrow_{\mathrm{p}\downarrow\mathrm{c}}$-normal form.

Definition. *$\longrightarrow_{Wk}$ is defined to be the union of $(Weak2)$, $(WAbs)$, $(WApp1)$, $(WApp2)$, $(WSubs)$, $(Cross)$, and $(Merge)$.*

Example 44 ($\xrightarrow[\mathrm{D}]{}$ simulation). Let

$$M \equiv v\langle v := x\langle y := zq\rangle\langle z := p\rangle\rangle \xrightarrow[\mathrm{D}]{} v\langle v := x\langle z := p\rangle\rangle \equiv N$$

with all variables distinct.

We must explicitly state the translation indexing now. The index set must at least contain the free variables of M, so we choose it to be $X = \{x, q, p\}$. We partition the index set into two sets, $Y = \{x, p\}$ and $Z = \{q\}$ such that Y contains free variables of M, occurrences of which are discarded in the reduction but which persist in N and Z contains the free variables of M which do not occur in N (the lost variables). The sequence begins as below.

$$\begin{array}{rl}
 & \mathcal{T}(M)_X \\
\equiv & C_X^{X_1,X_2}\big(W_{v'}\big(v\langle v := C_{p_1}^{p_3 p_4}(W_{z'}(W_y(x_1)\langle y := z\langle z := p_3\rangle q_1\rangle)\langle z' := p_4\rangle))\big) \\
 & \qquad \langle v' := C_{p_2}^{p_5 p_6}(W_{z'}(W_y(x_2)\langle y := z\langle z := p_5\rangle q_2\rangle)\langle z' := p_6\rangle))\big) \\
\xrightarrow[Weak1]{2} & C_q^{q_1,q_2} C_{xp}^{x_1 p_1, x_2 p_2}\big(W_{v'}\big(v\langle v := C_{p_1}^{p_3 p_4}(W_{z'}(W_{p_3 q_1}(x_1))\langle z' := p_4\rangle))\big) \\
 & \qquad \langle v' := C_{p_2}^{p_5 p_6}(W_{z'}(W_{p_5 q_2}(x_2))\langle z' := p_6\rangle))\big) \\
\xrightarrow[Wk]{} & C_q^{q_1,q_2} W_{q_1 q_2} C_{xp}^{x_1 p_1, x_2 p_2}\big(W_{v'}\big(v\langle v := C_{p_1}^{p_3 p_4}(W_{z'}(W_{p_3}(x_1))\langle z' := p_4\rangle))\big) \\
 & \qquad \langle v' := C_{p_2}^{p_5 p_6}(W_{z'}(W_{p_5}(x_2))\langle z' := p_6\rangle))\big) \\
\xrightarrow[Merge]{} & W_q C_{xp}^{x_1 p_1, x_2 p_2}\big(W_{v'}\big(v\langle v := C_{p_1}^{p_3 p_4}(W_{z'}(W_{p_3}(x_1))\langle z' := p_4\rangle))\big) \\
 & \qquad \langle v' := C_{p_2}^{p_5 p_6}(W_{z'}(W_{p_5}(x_2))\langle z' := p_6\rangle))\big)
\end{array}$$

We start by simulating the Λ_{sub} garbage collection with some $\longrightarrow_{Weak1}$ reductions, one for each copy of the discarded substitution that the translation introduces. This garbage collection generally creates weakenings which are then pulled upwards through the term, either to top-level or until they merge with contractions. For our demonstration, we first choose to pull up the weakenings W_{q_1} and W_{q_2} corresponding to the free variable q lost in the Λ_{sub} reduction. After the merge, the weakening W_q may be pulled up to the top of the term.

$$
\begin{array}{rl}
 & W_q C_{xp}^{x_1 p_1, x_2 p_2}\left(W_{v'}\left(v\langle v := C_{p_1}^{p_3 p_4}(W_{z'}(W_{p_3}(x_1))\langle z' := p_4\rangle)\rangle\right)\right. \\
 & \qquad\qquad \left.\langle v' := C_{p_2}^{p_5 p_6}(W_{z'}(W_{p_5}(x_2))\langle z' := p_6\rangle)\rangle\right) \\
\xrightarrow[Wk]{}\!\!\!\!\!\!\rightarrow & W_q C_{xp}^{x_1 p_1, x_2 p_2}\left(W_{v'}\left(v\langle v := C_{p_1}^{p_3 p_4} W_{p_3}(W_{z'}(x_1)\langle z' := p_4\rangle)\rangle\right)\right. \\
 & \qquad\qquad \left.\langle v' := C_{p_2}^{p_5 p_6} W_{p_5}(W_{z'}(x_2)\langle z' := p_6\rangle)\rangle\right) \\
\xrightarrow[Merge]{}\!\!\!\!\!\!\rightarrow & W_q C_{xp}^{x_1 p_1, x_2 p_2}\left(W_{v'}\left(v\langle v := W_{z'}(x_1)\langle z' := p_1\rangle\rangle\right)\langle v' := W_{z'}(x_2)\langle z' := p_2\rangle\rangle\right) \\
\equiv & \mathcal{T}(N)_X
\end{array}
$$

Next, we pull up the weakenings W_{p_3} and W_{p_5} corresponding to the variable p which is discarded in the $\longrightarrow_{Weak1}$ reductions but which occurs freely above the discarded subsitution in N. After more merges, the final term is equivalent to $\mathcal{T}(N)_X$.

Examples 41 to 44 hint at how substitutions are copied in the translation of a term depending on their level of nesting in other substitutions. In the *encoding* $\mathcal{A}(M)$ of a term, there will be 2^{g+d} copies of a substitution $\langle x := P\rangle$, where $g = 0$ if the substitution is garbage or $g = 1$ otherwise and d is the number of non-garbage subsitutions that $\langle x := P\rangle$ is contained inside (not under). However in a *translation* $\mathcal{T}(M)$ of a term, the matter is further complicated as substitutions are themselves duplicated by the $\longrightarrow_{Comp}$ rule. For this reason, the following proof concentrates on one copy of a redex in a translated term.

Proposition 45. *If* $M \xrightarrow[\mathrm{ACD}]{} N$ *then* $\mathcal{T}(M)_X \longrightarrow^{+}_{\lambda\mathrm{blxr}} \mathcal{T}(N)_X$.

Proof. Proof by case split. We write $\langle\Theta := \overrightarrow{T}\rangle$ to denote a sequence of nested substitutions $\langle y_1 := T_1\rangle \ldots \langle y_n := T_n\rangle$ and drop the indexing in the translation except for the $\xrightarrow[\mathrm{D}]{}$ case. Figures 2.2–2.5 depict the reduction graphs corresponding to the cases, where n is the number of copies of the redex generated by the translation as explained above.

Case $M \equiv C[(\lambda x.P)Q] \xrightarrow[\mathrm{A}]{} C[P\langle x := Q\rangle] \equiv N$**,** $x \in \mathrm{fv}(P)$

Figure 2.8 displays the general term[15] for $\mathcal{T}(M)$ followed by a series of reductions. Figure 2.11 displays the general term for $\mathcal{T}(N)$. The path $\mathcal{A}(N) \longrightarrow^{*}_{\mathrm{p}\downarrow\mathrm{c}} N'$ involves pushing all substitutions in besides the substitutions which arise from $\langle x := Q\rangle$. The renaming R_Y^X arises from the encoding. The substitutions with binders Φ' bind variables in encodings of both P and Q, the substitutions with binders Ξ_P and Θ_Q bind variables of P or Q respectively. We do not label the context $D - \mathrm{fv}((\lambda x.P)Q) = \mathrm{fv}(P\langle x := Q\rangle)$ and the same holds for the encodings by Lemma 34.1.

We must show that the final terms in both figures are equivalent. This can be shown by proving that $Q_1 \equiv Q_1'''$ and $Q_2 \equiv Q_2'''$. These can easily be shown by unwrapping their definitions and renaming variables. For example,

$$
\begin{array}{rl}
 & Q_2 \\
\equiv & R^{\Pi_1\Pi_2\Pi_3\Pi_4}_{\Psi_1\Psi_2\Psi_3\Psi_4}(\downarrow_{\mathrm{p}\downarrow\mathrm{c}}(\downarrow_{\mathrm{p}\downarrow\mathrm{c}}(R^{\Phi_1\Phi' X}_{\Pi_1\Pi' Y}(\mathcal{A}(Q))\langle\Theta_Q := \overrightarrow{S_Q}\rangle)\langle\Pi' := \overrightarrow{T_\Pi}\rangle)) \\
\equiv & R^{\Pi_2\Pi_3}_{\Psi_2\Psi_3}(\downarrow_{\mathrm{p}\downarrow\mathrm{c}}(\downarrow_{\mathrm{p}\downarrow\mathrm{c}}(R^{\Phi_1\Phi'\Pi_4 X}_{\Psi_1\Pi'\Psi_4 Y}(\mathcal{A}(Q))\langle\Theta_Q := \overrightarrow{S_Q}\rangle)\langle\Pi' := \overrightarrow{T_\Pi}\rangle)) \\
\equiv & R^{\Pi_2\Pi_3}_{\Psi_2\Psi_3}(\downarrow_{\mathrm{p}\downarrow\mathrm{c}}(\downarrow_{\mathrm{p}\downarrow\mathrm{c}}(R^{\Phi_1\Phi'\Pi_4 X}_{\Psi_1\Pi'\Psi_4 Y}(\mathcal{A}(Q))\langle\Theta_Q := \overrightarrow{S_Q}\rangle)\langle\Pi' := R^{\Phi_2}_{\Pi_2}(\overrightarrow{T})\rangle)) \\
\equiv & \downarrow_{\mathrm{p}\downarrow\mathrm{c}}(\downarrow_{\mathrm{p}\downarrow\mathrm{c}}(R^{\Phi_1\Phi'\Pi_4 X}_{\Psi_1\Pi'\Psi_4 Y}(\mathcal{A}(Q))\langle\Theta_Q := \overrightarrow{S_\Psi}\rangle)\langle\Pi' := \overrightarrow{T_\Psi}\rangle) \\
\equiv_\alpha & \downarrow_{\mathrm{p}\downarrow\mathrm{c}}(\downarrow_{\mathrm{p}\downarrow\mathrm{c}}(R^{\Phi_1\Phi'\Theta_Q\Pi_4 X}_{\Psi_1\Psi'\Theta_\Psi\Psi_4 Y}(\mathcal{A}(Q))\langle\Theta_\Psi := \overrightarrow{S_\Psi}\rangle)\langle\Psi' := \overrightarrow{T_\Psi}\rangle) \\
\equiv & Q_2'''
\end{array}
$$

[15] Well, almost. We have omitted some contractions concerning free variables in the substitutions with binders Φ', Ξ_P, and Θ_Q for clarity but Proposition 35 can justify this decision.

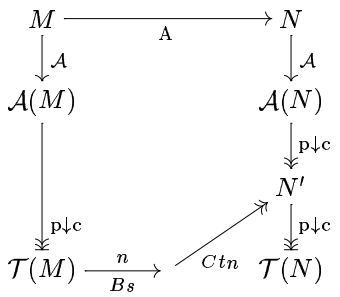


Figure 2.2: Reduction diagram for simulating $\longrightarrow_{\mathrm{A}}$ which does not create garbage

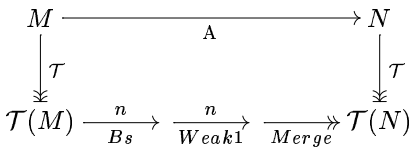


Figure 2.3: Reduction diagram for simulating $\longrightarrow_{\mathrm{A}}$ which does create garbage

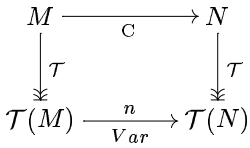


Figure 2.4: Reduction diagram for simulating $\longrightarrow_{\mathrm{C}}$

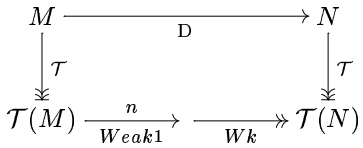


Figure 2.5: Reduction diagram for simulating $\longrightarrow_{\mathrm{D}}$

Case $M \equiv C[(\lambda x.P)Q] \xrightarrow[\mathrm{A}]{} C[P\langle x := Q\rangle] \equiv N$**,** $x \notin \mathrm{fv}(P)$

This case is similar to the last one. Figure 2.14 displays the general term for $\mathcal{T}(M)$ followed by a series of reductions. Figure 2.17 displays the general term for $\mathcal{T}(N)$. The renaming R^X_Y and the substitutions with binders Φ', Ξ_P, and Θ_Q arise as before.

Case $M \equiv C[C'[x]\langle x := Q\rangle] \xrightarrow[\mathrm{C}]{} C[C'[Q]\langle x := Q\rangle] \equiv N$**.**

We treat the case where the the term x to be replaced is not the only free occurrence of x in $C_2[x]$. The reduction sequences are shown in Figures 2.19 and 2.20.

The two underlined subterms in those figures reduce to almost the same term – in both, the substitution with binder x is distributed through the subterms. Both subterms are subterms of the translations

$$C^{\Pi,\Psi_1}_{\Phi} C^{\Gamma_2,\Psi_2}_{\Theta\setminus\Phi}(W_{x'}(\downarrow_{\mathrm{p}\downarrow\mathrm{c}}(C^{\Delta'_1,\Gamma'_1}_{\Pi'}(D_3[x_i\langle x_i := P_i\rangle)\rangle])))\langle x' := P_2\rangle)$$

and

$$C^{\Pi_1,\Psi_1}_{\Phi} C^{\Pi_2,\Psi_2}_{\Theta\setminus\Phi}(W'_x(\downarrow_{\mathrm{p}\downarrow\mathrm{c}}(C^{\Delta_1,\Gamma_1}_{\Pi_1} C^{\Delta_2,\Gamma_2}_{\Pi_2}(D'_2[P'_i]\langle x := P'_1\rangle)))\langle x' := P_2\rangle)$$

of Λ_{sub} terms. Ignoring linearity and contractions for a moment, the two terms above are identical except that one has a subterm P where the other has $x\langle x := P\rangle$. By Proposition 35, the contractions of both

$$\mathrm{fv}(R^X_Y(Q)) = \Phi_1 \uplus \Phi' \uplus \Theta_Q \uplus \Pi_4$$
$$\mathrm{fv}(R^X_Y(P)) = \Phi_1 \uplus \Phi' \uplus \Xi_P \uplus W$$
$$\mathrm{fv}(\overrightarrow{S_Q}) = \Pi_3 \quad \mathrm{fv}(\overrightarrow{T}) = \Phi_2 \quad \overrightarrow{T_\Delta} = R^{\Phi_2}_{\Delta_2}(\overrightarrow{T}) \quad \overrightarrow{T_\Pi} = R^{\Phi_2}_{\Pi_2}(\overrightarrow{T})$$

Figure 2.6: Sets of free variables

1. $P' = R^{\Phi_1\Phi' X}_{\Delta_1\Delta' Y}(\mathcal{A}(P)) \quad Q' = R^{\Phi_1\Phi' X}_{\Pi_1\Pi' Y}(\mathcal{A}(Q))$
2. $P'' = {\downarrow_{\mathrm{p}\downarrow\mathrm{c}}}(P'\langle \Xi_P := \overrightarrow{S_P}\rangle) \quad Q'' = {\downarrow_{\mathrm{p}\downarrow\mathrm{c}}}(Q'\langle \Theta_Q := \overrightarrow{S_Q}\rangle)$
3. $P''' = {\downarrow_{\mathrm{p}\downarrow\mathrm{c}}}(P''\langle \Delta' := \overrightarrow{T_\Delta}\rangle) \quad Q''' = {\downarrow_{\mathrm{p}\downarrow\mathrm{c}}}(Q''\langle \Pi' := \overrightarrow{T_\Pi}\rangle)$
4. $Q_1 = R^{\Pi_1\Pi_2\Pi_3\Pi_4}_{\Gamma_1\Gamma_2\Gamma_3\Gamma_4}(Q''') \quad Q_2 = R^{\Pi_1\Pi_2\Pi_3\Pi_4}_{\Psi_1\Psi_2\Psi_3\Psi_4}(Q''')$

Figure 2.7: Abbreviations in Figure 2.8

$$\begin{array}{rl}
 & \mathcal{T}(C[(\lambda x.P)Q]) \\
\equiv & {\downarrow_{\mathrm{p}\downarrow\mathrm{c}}}(\mathcal{A}(C)[\, R^X_Y(\mathcal{A}((\lambda x.P)Q))]) \\
 & (1) \\
\equiv & D[{\downarrow_{\mathrm{p}\downarrow\mathrm{c}}}(C^{\Delta_1,\Pi_1}_{\Phi_1} C^{\Delta',\Pi'}_{\Phi'}((\lambda x.P')Q')\langle \Phi' := \overrightarrow{T}\rangle\langle \Xi_P := \overrightarrow{S_P}\rangle\langle \Theta_Q := \overrightarrow{S_Q}\rangle)] \\
 & (2) \\
\equiv & D[{\downarrow_{\mathrm{p}\downarrow\mathrm{c}}}(C^{\Delta_1,\Pi_1}_{\Phi_1} C^{\Delta',\Pi'}_{\Phi'}((\lambda x.P'')Q'')\langle \Phi' := \overrightarrow{T}\rangle)] \\
\equiv & D[C^{\Delta_1,\Pi_1}_{\Phi_1} C^{\Delta_2,\Pi_2}_{\Phi_2}((\lambda x.\, {\downarrow_{\mathrm{p}\downarrow\mathrm{c}}}(P''\langle \Delta' := \overrightarrow{T_\Delta}\rangle))\, {\downarrow_{\mathrm{p}\downarrow\mathrm{c}}}(Q''\langle \Pi' := \overrightarrow{T_\Pi}\rangle))] \\
 & (3) \\
\equiv & D[C^{\Delta_1\Delta_2,\Pi_1\Pi_2}_{\Phi_1\Phi_2}((\lambda x.P''')Q''')] \\
 & (4) \\
\xrightarrow[Bs]{} & D[C^{\Delta_1\Delta_2,\Pi_1\Pi_2}_{\Phi_1\Phi_2} C^{\Gamma_1\Gamma_2,\Psi_1\Psi_2}_{\Pi_1\Pi_2} C^{\Gamma_3\Gamma_4,\Psi_3\Psi_4}_{\Pi_3\Pi_4}(W_{x'}(P'''\langle x := Q_1\rangle)\langle x' := Q_2\rangle)] \\
\xrightarrow[Ctn]{}\!\!\!\!\!\!\rightarrow & D[C^{\Pi_1\Pi_2,\Psi_1\Psi_2}_{\Phi_1\Phi_2} C^{\Gamma_3\Gamma_4,\Psi_3\Psi_4}_{\Pi_3\Pi_4}(W_{x'}(C^{\Delta_1\Delta_2,\Gamma_1\Gamma_2}_{\Pi_1\Pi_2}(P'''\langle x := Q_1\rangle)\langle x' := Q_2\rangle) \\
\longrightarrow^{+}_{\mathrm{p}\downarrow\mathrm{c}} & D[C^{\Pi_1\Pi_2,\Psi_1\Psi_2}_{\Phi_1\Phi_2} C^{\Gamma_3\Gamma_4,\Psi_3\Psi_4}_{\Pi_3\Pi_4}(W_{x'}({\downarrow_{\mathrm{p}\downarrow\mathrm{c}}}(C^{\Delta_1\Delta_2,\Gamma_1\Gamma_2}_{\Pi_1\Pi_2}(P'''\langle x := Q_1\rangle)))\langle x' := Q_2\rangle)]
\end{array}$$

Figure 2.8: Translation for $C[(\lambda x.P)Q]$, x a free variable of P

$$\begin{array}{l}
\mathrm{fv}(R^X_Y(Q)) = \Phi_1 \uplus \Phi' \uplus \Theta_Q \uplus \Pi_4 \\
\mathrm{fv}(R^X_Y(P)) = \Phi_1 \uplus \Phi' \uplus \Xi_P \uplus W \\
\mathrm{fv}(\overrightarrow{S_Q}) = \Pi_3 \quad \overrightarrow{S_\Gamma} = R^{\Pi_3}_{\Gamma_3}(\overrightarrow{S_Q}) \quad \overrightarrow{S_\Psi} = R^{\Pi_3}_{\Psi_3}(\overrightarrow{S_Q}) \\
\mathrm{fv}(\overrightarrow{T}) = \Phi_2 \quad \overrightarrow{T_\Pi} = R^{\Phi_2}_{\Pi_2}(\overrightarrow{T}) \quad \overrightarrow{T_\Psi} = R^{\Phi_2}_{\Psi_2}(\overrightarrow{T}) \quad \overrightarrow{T_\Gamma} = R^{\Phi_2}_{\Gamma_2}(\overrightarrow{T}) \quad \overrightarrow{T_\Delta} = R^{\Phi_2}_{\Delta_2}(\overrightarrow{T})
\end{array}$$

Figure 2.9: Sets of free variables

1. $P' = R^{\Phi_1\Phi' X}_{\Delta_1\Delta' Y}(\mathcal{A}(P)) \quad Q'_1 = R^{\Phi_1\Phi'\Theta_Q\Pi_4 X}_{\Gamma_1\Gamma'\Theta_\Gamma\Gamma_4 Y}(\mathcal{A}(Q)) \quad Q'_2 = R^{\Phi_1\Phi'\Theta_Q\Pi_4 X}_{\Psi_1\Psi'\Theta_\Psi\Psi_4 Y}(\mathcal{A}(Q))$
2. $P'' = {\downarrow_{\mathrm{p}\downarrow\mathrm{c}}}(P'\langle \Xi_P := \overrightarrow{S_P}\rangle)$
3. $Q''_1 = {\downarrow_{\mathrm{p}\downarrow\mathrm{c}}}(Q'_1\langle \Theta_\Gamma := \overrightarrow{S_\Gamma}\rangle) \quad Q''_2 = {\downarrow_{\mathrm{p}\downarrow\mathrm{c}}}(Q'_2\langle \Theta_\Psi := \overrightarrow{S_\Psi}\rangle))$
4. $P''' = {\downarrow_{\mathrm{p}\downarrow\mathrm{c}}}(P''\langle \Delta' := \overrightarrow{T_\Delta}\rangle) \quad Q'''_1 = {\downarrow_{\mathrm{p}\downarrow\mathrm{c}}}(Q''_1\langle \Gamma' := \overrightarrow{T_\Gamma}\rangle) \quad Q'''_2 = {\downarrow_{\mathrm{p}\downarrow\mathrm{c}}}(Q''_2\langle \Psi' := \overrightarrow{T_\Psi}\rangle)$

Figure 2.10: Abbreviations in Figure 2.11

$$\begin{array}{ll}
 & \mathcal{T}(C[P\langle x := Q\rangle]) \\
\equiv & {\downarrow_{\mathrm{p}\downarrow\mathrm{c}}}(\mathcal{A}(C)[\, R^X_Y(\mathcal{A}(P\langle x := Q\rangle))]) \\
 & (1) \\
\equiv & D[{\downarrow_{\mathrm{p}\downarrow\mathrm{c}}}(C^{\Pi_1\Pi',\Psi_1\Psi'}_{\Phi_1\Phi'} C^{\Theta_\Gamma\Gamma_4,\Theta_\Psi\Psi_4}_{\Theta_Q\Pi_4}(W_{x'}(C^{\Delta_1\Delta',\Gamma_1\Gamma'}_{\Pi_1\Pi'}(P'\langle x := Q'_1\rangle))\langle x' := Q'_2\rangle))\langle \Phi' := \overrightarrow{T}\rangle\langle \Xi_P := \overrightarrow{S_P}\rangle\langle \Theta_Q := \overrightarrow{S_Q}\rangle]) \\
 & (2) \\
\equiv & D[{\downarrow_{\mathrm{p}\downarrow\mathrm{c}}}(C^{\Pi_1\Pi',\Psi_1\Psi'}_{\Phi_1\Phi'} C^{\Gamma_3\Gamma_4,\Psi_3\Psi_4}_{\Pi_3\Pi_4}(W_{x'}(C^{\Delta_1\Delta',\Gamma_1\Gamma'}_{\Pi_1\Pi'}(P''\langle x := Q'_1\langle \Theta_\Gamma := \overrightarrow{S_\Gamma}\rangle\rangle))\langle x' := Q'_2\langle \Theta_\Psi := \overrightarrow{S_\Psi}\rangle\rangle))\langle \Phi' := \overrightarrow{T}\rangle]) \\
 & (3) \\
\equiv & D[{\downarrow_{\mathrm{p}\downarrow\mathrm{c}}}(C^{\Pi_1\Pi',\Psi_1\Psi'}_{\Phi_1\Phi'} C^{\Gamma_3\Gamma_4,\Psi_3\Psi_4}_{\Pi_3\Pi_4}(W_{x'}(C^{\Delta_1\Delta',\Gamma_1\Gamma'}_{\Pi_1\Pi'}(P''\langle x := Q''_1\rangle))\langle x' := Q''_2\rangle))\langle \Phi' := \overrightarrow{T}\rangle]) \\
\equiv & D[{\downarrow_{\mathrm{p}\downarrow\mathrm{c}}}(C^{\Pi_1\Pi_2,\Psi_1\Psi_2}_{\Phi_1\Phi_2} C^{\Gamma_3\Gamma_4,\Psi_3\Psi_4}_{\Pi_3\Pi_4}(W_{x'}(C^{\Delta_1\Delta',\Gamma_1\Gamma'}_{\Pi_1\Pi'}(P''\langle x := Q''_1\rangle)\langle \Pi' := \overrightarrow{T_\Pi}\rangle)\langle x' := Q''_2\langle \Psi' := \overrightarrow{T_\Psi}\rangle\rangle))]) \\
\equiv & D[{\downarrow_{\mathrm{p}\downarrow\mathrm{c}}}(C^{\Pi_1\Pi_2,\Psi_1\Psi_2}_{\Phi_1\Phi_2} C^{\Gamma_3\Gamma_4,\Psi_3\Psi_4}_{\Pi_3\Pi_4}(W_{x'}(C^{\Delta_1\Delta_2,\Gamma_1\Gamma_2}_{\Pi_1\Pi_2}({\downarrow_{\mathrm{p}\downarrow\mathrm{c}}}(P''\langle \Delta' := \overrightarrow{T_\Delta}\rangle)\langle x := {\downarrow_{\mathrm{p}\downarrow\mathrm{c}}}(Q''_1\langle \Gamma' := \overrightarrow{T_\Gamma}\rangle)\rangle))\langle x' := {\downarrow_{\mathrm{p}\downarrow\mathrm{c}}}(Q''_2\langle \Psi' := \overrightarrow{T_\Psi}\rangle)\rangle))]) \\
 & (4) \\
\equiv & D[C^{\Pi_1\Pi_2,\Psi_1\Psi_2}_{\Phi_1\Phi_2} C^{\Gamma_3\Gamma_4,\Psi_3\Psi_4}_{\Pi_3\Pi_4}(W_{x'}({\downarrow_{\mathrm{p}\downarrow\mathrm{c}}}(C^{\Delta_1\Delta_2,\Gamma_1\Gamma_2}_{\Pi_1\Pi_2}(P'''\langle x := Q'''_1\rangle)))\langle x' := Q'''_2\rangle)]
\end{array}$$

Figure 2.11: Translation for $C[P\langle x := Q\rangle]$, x a free variable of P

$$\mathrm{fv}(R_Y^X(Q)) = \Phi_1 \uplus \Phi' \uplus \Theta_Q \uplus \Pi_4$$
$$\mathrm{fv}(R_Y^X(P)) = \Phi_1 \uplus \Phi' \uplus \Xi_P \uplus W$$
$$\mathrm{fv}(\overrightarrow{S_Q}) = \Pi_3 \quad \mathrm{fv}(\overrightarrow{T}) = \Phi_2 \quad \overrightarrow{T_\Delta} = R_{\Delta_2}^{\Phi_2}(\overrightarrow{T}) \quad \overrightarrow{T_\Pi} = R_{\Pi_2}^{\Phi_2}(\overrightarrow{T})$$

Figure 2.12: Sets of free variables

1. $P' = R_{\Delta_1\Delta' Y}^{\Phi_1\Phi' X}(\mathcal{A}(P)) \quad Q' = R_{\Pi_1\Pi' Y}^{\Phi_1\Phi' X}(\mathcal{A}(Q))$
2. $P'' = \downarrow_{\mathrm{p}\downarrow\mathrm{c}}(P'\langle \Xi_P := \overrightarrow{S_P}\rangle) \quad Q'' = \downarrow_{\mathrm{p}\downarrow\mathrm{c}}(Q'\langle \Theta_Q := \overrightarrow{S_Q}\rangle)$
3. $P_1 = \downarrow_{\mathrm{p}\downarrow\mathrm{c}}(P''\langle \Delta' := \overrightarrow{T_\Delta}\rangle) \quad Q_1 = \downarrow_{\mathrm{p}\downarrow\mathrm{c}}(Q''\langle \Pi' := \overrightarrow{T_\Pi}\rangle)$

Figure 2.13: Abbreviations in Figure 2.14

$$\begin{array}{rl}
 & \mathcal{T}(C[(\lambda x.P)Q]) \\
\equiv & \downarrow_{\mathrm{p}\downarrow\mathrm{c}}(\mathcal{A}(C)[\, R_Y^X(\mathcal{A}((\lambda x.P)Q))]) \\
 & (1) \\
\equiv & D[\downarrow_{\mathrm{p}\downarrow\mathrm{c}}((C_{\Phi_1\Phi'}^{\Delta_1\Delta',\Pi_1\Pi'}((\lambda x.W_x(P'))Q'))\langle \Phi' := \overrightarrow{T}\rangle\langle \Xi_P := \overrightarrow{S_P}\rangle\langle \Theta_Q := \overrightarrow{S_Q}\rangle)] \\
 & (2) \\
\equiv & D[\downarrow_{\mathrm{p}\downarrow\mathrm{c}}((C_{\Phi_1\Phi'}^{\Delta_1\Delta',\Pi_1\Pi'}((\lambda x.W_x(P''))Q''))\langle \Phi' := \overrightarrow{T}\rangle)] \\
\equiv & D[C_{\Phi_1\Phi_2}^{\Delta_1\Delta_2,\Pi_1\Pi_2}((\lambda x.\downarrow_{\mathrm{p}\downarrow\mathrm{c}}(W_x(P''\langle \Delta' := \overrightarrow{T_\Delta}\rangle)))\downarrow_{\mathrm{p}\downarrow\mathrm{c}}(Q''\langle \Pi' := \overrightarrow{T_\Pi}\rangle))] \\
 & (3) \\
\equiv & D[C_{\Phi_1\Phi_2}^{\Delta_1\Delta_2,\Pi_1\Pi_2}((\lambda x.W_x(P_1))Q_1)] \\
\xrightarrow{Bs} & D[C_{\Phi_1\Phi_2}^{\Delta_1\Delta_2,\Pi_1\Pi_2}(C_{\mathrm{fv}(Q_1)}^{\Gamma,\Psi}(W_{x'}(W_x(P_1)\langle x := R_\Gamma^{\mathrm{fv}(Q_1)}(Q_1)\rangle)\langle x' := R_\Psi^{\mathrm{fv}(Q_1)}(Q_1)\rangle))] \\
\xrightarrow{Weak1} & D[C_{\Phi_1\Phi_2}^{\Delta_1\Delta_2,\Pi_1\Pi_2}(C_{\mathrm{fv}(Q_1)}^{\Gamma,\Psi}(W_\Psi(W_x(P_1)\langle x := R_\Gamma^{\mathrm{fv}(Q_1)}(Q_1)\rangle)))] \\
\xrightarrow{Merge} & D[C_{\Phi_1\Phi_2}^{\Delta_1\Delta_2,\Pi_1\Pi_2}(W_x(P_1)\langle x := Q_1\rangle)]
\end{array}$$

Figure 2.14: Translation for $C[P\langle x := Q\rangle]$, x not a free variable of P

$$\begin{array}{l}\mathrm{fv}(R^X_Y(Q)) = \Phi_1 \uplus \Phi' \uplus \Theta_Q \uplus \Pi_4\\ \mathrm{fv}(R^X_Y(P)) = \Phi_1 \uplus \Phi' \uplus \Xi_P \uplus W\\ \mathrm{fv}(\overrightarrow{T}) = \Phi_2 \quad \overrightarrow{T_\Delta} = R^{\Phi_2}_{\Delta_2}(\overrightarrow{T}) \quad \overrightarrow{T_\Pi} = R^{\Phi_2}_{\Pi_2}(\overrightarrow{T})\end{array}$$

Figure 2.15: Sets of free variables

1. $P' = R^{\Phi_1\Phi' X}_{\Delta_1\Delta' Y}(\mathcal{A}(P)) \quad Q' = R^{\Phi_1\Phi' X}_{\Pi_1\Pi' Y}(\mathcal{A}(Q))$
2. $P'' = {\downarrow_{\mathrm{p}\downarrow\mathrm{c}}}(P'\langle\Xi_P := \overrightarrow{S_P}\rangle) \quad Q'' = {\downarrow_{\mathrm{p}\downarrow\mathrm{c}}}(Q'\langle\Theta_Q := \overrightarrow{S_Q}\rangle)$
3. $P_1 = {\downarrow_{\mathrm{p}\downarrow\mathrm{c}}}(P''\langle\Delta' := \overrightarrow{T_\Delta}\rangle) \quad Q_1 = {\downarrow_{\mathrm{p}\downarrow\mathrm{c}}}(Q''\langle\Pi' := \overrightarrow{T_\Pi}\rangle)$

Figure 2.16: Abbreviations in Figure 2.17

$$\begin{array}{ll} & \mathcal{T}(C[P\langle x := Q\rangle])\\ \equiv & {\downarrow_{\mathrm{p}\downarrow\mathrm{c}}}(\mathcal{A}(C)[\,R^X_Y(\mathcal{A}(P\langle x := Q\rangle))])\\ & (1)\\ \equiv & D[{\downarrow_{\mathrm{p}\downarrow\mathrm{c}}}((C^{\Delta_1\Delta',\Pi_1\Pi'}_{\Phi_1\Phi'}(W_x(P')\langle x := Q'\rangle))\langle\Phi' := \overrightarrow{T}\rangle\langle\Xi_P := \overrightarrow{S_P}\rangle\langle\Theta_Q := \overrightarrow{S_Q}\rangle)]\\ & (2)\\ \equiv & D[{\downarrow_{\mathrm{p}\downarrow\mathrm{c}}}((C^{\Delta_1\Delta',\Pi_1\Pi'}_{\Phi_1\Phi'}(W_x(P'')\langle x := Q''\rangle))\langle\Phi' := \overrightarrow{T}\rangle)]\\ \equiv & D[(C^{\Delta_1\Delta_2,\Pi_1\Pi_2}_{\Phi_1\Phi_2}(W_x({\downarrow_{\mathrm{p}\downarrow\mathrm{c}}}(P''\langle\Delta' := \overrightarrow{T_\Delta}\rangle))\langle x := {\downarrow_{\mathrm{p}\downarrow\mathrm{c}}}(Q''\langle\Pi' := \overrightarrow{T_\Pi}\rangle)\rangle))))]\\ & (3)\\ \equiv & D[C^{\Delta_1\Delta_2,\Pi_1\Pi_2}_{\Phi_1\Phi_2}(W_x(P_1)\langle x := Q_1\rangle)]\end{array}$$

Figure 2.17: Translation for $C[P\langle x := Q\rangle]$, x not a free variable of P

$$\begin{aligned}&\mathrm{fv}(P) = \Theta\\ &P_2 = R^{\Phi\ \ \Theta\setminus\Phi}_{\Psi_1\ \Psi_2}(\mathcal{A}(P))\end{aligned}$$

Figure 2.18: Free variables and abbreviations for Figures 2.19 and 2.20

$$\begin{aligned}&\mathcal{T}(C_1[C_2[x]\langle x := P\rangle])\\ \equiv\ &\downarrow_{\mathrm{p}\downarrow\mathrm{c}}(\mathcal{A}(C_1)[\,R^W_X(\mathcal{A}(C_2[x]\langle x := P\rangle))])\\ \equiv\ &\downarrow_{\mathrm{p}\downarrow\mathrm{c}}(D_1[R^W_X((C^{\Pi_1,\Psi_1}_{\Phi} C^{\Gamma_2,\Psi_2}_{\Theta\setminus\Phi}(W_{x'}(C^{\Delta_1,\Gamma_1}_{\Pi_1}(D_2[x_i]\langle x := P_1\rangle))\langle x' := P_2\rangle))])\\ \equiv\ &\downarrow_{\mathrm{p}\downarrow\mathrm{c}}(D_1[R^W_X(C^{\Pi_1,\Psi_1}_{\Phi} C^{\Gamma_2,\Psi_2}_{\Theta\setminus\Phi}(W_{x'}(C^{\Delta'_1,\Gamma'_1}_{\Pi'_1}(D_3[x_i\langle x_i := P_i\rangle])\rangle))\langle x' := P_2\rangle))])\\ \equiv\ &\downarrow_{\mathrm{p}\downarrow\mathrm{c}}(D_1[R^W_X(C^{\Pi,\Psi_1}_{\Phi} C^{\Gamma_2,\Psi_2}_{\Theta\setminus\Phi}(W_{x'}(\underline{\downarrow_{\mathrm{p}\downarrow\mathrm{c}}(C^{\Delta'_1,\Gamma'_1}_{\Pi'}(D_3[x_i\langle x_i := P_i\rangle])\rangle)]))\langle x' := P_2\rangle))])\\ \equiv\ &\downarrow_{\mathrm{p}\downarrow\mathrm{c}}(D_1[R^W_X(C^{\Pi,\Psi_1}_{\Phi} C^{\Gamma_2,\Psi_2}_{\Theta\setminus\Phi}(W_{x'}(D[x_i\langle x_i := P_i\rangle])\langle x' := P_2\rangle))])\\ \equiv\ &E[x_i\langle x_i := P_i\rangle]\end{aligned}$$

Figure 2.19: Translation for $C_1[C_2[x]\langle x := P\rangle]$

$$\begin{aligned}&\mathcal{T}(C_1[C'_2[P]\langle x := P\rangle])\\ \equiv\ &\downarrow_{\mathrm{p}\downarrow\mathrm{c}}(\mathcal{A}(C_1)[\,R^W_X(\mathcal{A}(C'_2[P]\langle x := P\rangle))])\\ \equiv\ &\downarrow_{\mathrm{p}\downarrow\mathrm{c}}(D_1[R^W_X(C^{\Pi_1,\Psi_1}_{\Phi} C^{\Pi_2,\Psi_2}_{\Theta\setminus\Phi}(W'_x(C^{\Delta_1,\Gamma_1}_{\Pi_1} C^{\Delta_2,\Gamma_2}_{\Pi_2}(D'_2[P'_i]\langle x := P'_1\rangle))\langle x' := P_2\rangle))])\\ \equiv\ &\downarrow_{\mathrm{p}\downarrow\mathrm{c}}(D_1[R^W_X(C^{\Pi_1,\Psi_1}_{\Phi} C^{\Pi_2,\Psi_2}_{\Theta\setminus\Phi}(W'_x(\underline{\downarrow_{\mathrm{p}\downarrow\mathrm{c}}(C^{\Delta_1,\Gamma_1}_{\Pi_1} C^{\Delta_2,\Gamma_2}_{\Pi_2}(D'_2[P'_i]\langle x := P'_1\rangle))})\langle x' := P_2\rangle))])\\ \equiv\ &\downarrow_{\mathrm{p}\downarrow\mathrm{c}}(D_1[R^W_X(C^{\Pi_1,\Psi_1}_{\Phi} C^{\Pi_2,\Psi_2}_{\Theta\setminus\Phi}(W'_x(D[P_i])\langle x' := P_2\rangle))])\\ \equiv\ &E[P_i]\end{aligned}$$

Figure 2.20: Translation for $C_1[C_2[P]\langle x := P\rangle]$, x a free variable of $C_2[P]$

$$\begin{aligned}&\mathcal{T}(C_1[C'_2[P]\langle x := P\rangle])\\ \equiv\ &\downarrow_{\mathrm{p}\downarrow\mathrm{c}}(\mathcal{A}(C_1)[\,R^W_X(\mathcal{A}(C'_2[P]\langle x := P\rangle))])\\ \equiv\ &\downarrow_{\mathrm{p}\downarrow\mathrm{c}}(D_1[R^W_X(C^{\Pi_1,\Psi_1}_{\Phi} C^{\Pi_2,\Psi_2}_{\Theta\setminus\Phi}(W_x(D'_2[P'_i])\langle x := P_2\rangle))])\\ \equiv\ &\downarrow_{\mathrm{p}\downarrow\mathrm{c}}(D_1[R^W_X(C^{\Pi_1,\Psi_1}_{\Phi} C^{\Pi_2,\Psi_2}_{\Theta\setminus\Phi}(W_x(\underline{\downarrow_{\mathrm{p}\downarrow\mathrm{c}}(D'_2[P'_i])})\langle x := P_2\rangle))])\\ \equiv\ &\downarrow_{\mathrm{p}\downarrow\mathrm{c}}(D_1[R^W_X(C^{\Pi_1,\Psi_1}_{\Phi} C^{\Pi_2,\Psi_2}_{\Theta\setminus\Phi}(W_x(D[P_i])\langle x := P_2\rangle))])\\ \equiv\ &E[P_i]\end{aligned}$$

Figure 2.21: Translation for $C_1[C_2[P]\langle x := P\rangle]$, x not a free variable of $C_2[P]$

$W \subseteq \mathrm{fv}(P) \quad Y \subseteq \mathrm{fv}(Q) \setminus \mathrm{fv}(P)$
$\mathrm{fv}(R^{WY}_{X\,Z}(P)) = \mathrm{fv}(R^{W}_{X}(P)) = \Phi_1 \uplus \Phi' \uplus \Xi_P \uplus W$
$\mathrm{fv}(R^{WY}_{X\,Z}(Q)) = \Phi_1 \uplus \Phi' \uplus \Theta_Q \uplus \Pi_4 \uplus \Pi_5$
$Y \subseteq \Theta_Q \uplus \Pi_4 \uplus \Pi_5$
$D[\ ] = W_{U \setminus \mathrm{fv}(M)}[\mathcal{T}(C[\ ])_\emptyset]$
V is the set of free variables of Q not bound in M or not occuring free above the hole in $C[\ ]$ or in P (the lost variables)
$D'[\ ] = W_V D_2[\ ]$
$C'[\ ] = C[\ ]$ except that the free variables of the hole in $C'[\ ]$ do not contain V
$\Pi_5 = R^{Y}_{Z}(V)$
$\mathrm{fv}(\overrightarrow{S_Q}) = \Pi_3 \quad \mathrm{fv}(\overrightarrow{T}) = \Phi_2 \quad \overrightarrow{T_\Delta} = R^{\Phi_2}_{\Delta_2}(\overrightarrow{T}) \quad \overrightarrow{T_\Pi} = R^{\Phi_2}_{\Pi_2}(\overrightarrow{T})$

Figure 2.22: Free variables and abbreviations for Figures 2.23 and 2.24

$$
\begin{array}{ll}
 & \mathcal{T}(C[P\langle x := Q\rangle])_U \\
\equiv & \downarrow_{\mathrm{p}\downarrow\mathrm{c}}(\mathcal{A}(C)_U[\,R^{WY}_{X\,Z}(\mathcal{A}(P\langle x := Q\rangle))]) \\
\equiv & D[\downarrow_{\mathrm{p}\downarrow\mathrm{c}}((C^{\Delta_1\Delta',\Pi_1\Pi'}_{\Phi_1\Phi'}(W_x(\mathcal{A}(R^{\Phi_1\,\Phi'\,W}_{\Delta_1\,\Delta'\,X}(P)))\langle x := \mathcal{A}(R^{\Phi_1\,\Phi'\,WY}_{\Pi_1\,\Pi'\,X\,Z}(Q)))))\langle\Phi' := \overrightarrow{T}\rangle\langle\Xi_P := \overrightarrow{S_P}\rangle\langle\Theta_Q := \overrightarrow{S_Q}\rangle)] \\
\equiv & D[C^{\Delta_1\Delta_2,\Pi_1\Pi_2}_{\Phi_1\Phi_2}(W_x(\downarrow_{\mathrm{p}\downarrow\mathrm{c}}(\mathcal{A}(R^{\Phi_1\,\Phi'\,W}_{\Delta_1\,\Delta'\,X}(P))\langle\Delta' := \overrightarrow{T_\Delta}\rangle\langle\Xi_P := \overrightarrow{S_P}\rangle))\langle x := \downarrow_{\mathrm{p}\downarrow\mathrm{c}}(\mathcal{A}(R^{\Phi_1\,\Phi'\,WY}_{\Pi_1\,\Pi'\,X\,Z}(Q))\langle\Pi' := \overrightarrow{T_\Pi}\rangle\langle\Theta_Q := \overrightarrow{S_Q}\rangle)))] \\
\overrightarrow{Weak1} & D[C^{\Delta_1\Delta_2,\Pi_1\Pi_2}_{\Phi_1\Phi_2}(W_{\Pi_1\Pi_2\Pi_3\Pi_4\Pi_5}(\downarrow_{\mathrm{p}\downarrow\mathrm{c}}(\mathcal{A}(R^{\Phi_1\,\Phi'\,W}_{\Delta_1\,\Delta'\,X}(P))\langle\Delta' := \overrightarrow{T_\Delta}\rangle\langle\Xi_P := \overrightarrow{S_P}\rangle)))] \\
\overrightarrow{\overrightarrow{Merge}} & D[R^{\Delta_1\Delta_2}_{\Phi_1\Phi_2}(W_{\Pi_3\Pi_4\Pi_5}(\downarrow_{\mathrm{p}\downarrow\mathrm{c}}(\mathcal{A}(R^{\Phi_1\,\Phi'\,W}_{\Delta_1\,\Delta'\,X}(P))\langle\Delta' := \overrightarrow{T_\Delta}\rangle\langle\Xi_P := \overrightarrow{S_P}\rangle)))] \\
\equiv & D[W_{\Pi_3\Pi_4\Pi_5}(\downarrow_{\mathrm{p}\downarrow\mathrm{c}}(\mathcal{A}(R^{\Phi'\,W}_{\Delta'\,X}(P))\langle\Delta' := \overrightarrow{T}\rangle\langle\Xi_P := \overrightarrow{S_P}\rangle))] \\
\equiv_\alpha & D[W_{\Pi_3\Pi_4\Pi_5}(\downarrow_{\mathrm{p}\downarrow\mathrm{c}}(\mathcal{A}(R^{W}_{X}(P))\langle\Phi' := \overrightarrow{T}\rangle\langle\Xi_P := \overrightarrow{S_P}\rangle))] \\
\equiv & M_1 \\
\overrightarrow{\overrightarrow{Wk}} & W_V D_1[W_{\Pi_3\Pi_4}(\downarrow_{\mathrm{p}\downarrow\mathrm{c}}(\mathcal{A}(R^{W}_{X}(P))\langle\Phi' := \overrightarrow{T}\rangle\langle\Xi_P := \overrightarrow{S_P}\rangle))] \\
\equiv & M_2 \\
\overrightarrow{\overrightarrow{Wk}} & M_3
\end{array}
$$

Figure 2.23: Translation for $C[P\langle x := Q\rangle]$, x not a free variable of P

$$
\begin{array}{ll}
 & \mathcal{T}(C'[P])_U \\
\equiv & \downarrow_{\mathrm{p}\downarrow\mathrm{c}}(\mathcal{A}(C)_U[\,R^{W}_{X}(\mathcal{A}(P))]) \\
\equiv & D'[\downarrow_{\mathrm{p}\downarrow\mathrm{c}}(\mathcal{A}(R^{W}_{X}(P))\langle\Phi' := \overrightarrow{T}\rangle\langle\Xi_P := \overrightarrow{S_P}\rangle)] \\
\equiv & W_V D_2[\downarrow_{\mathrm{p}\downarrow\mathrm{c}}(\mathcal{A}(R^{W}_{X}(P))\langle\Phi' := \overrightarrow{T}\rangle\langle\Xi_P := \overrightarrow{S_P}\rangle)]
\end{array}
$$

Figure 2.24: Translation for $C[P]$

terms are at their most efficient. Therefore, we can write both underlined subterms with the same context D as $D[P_i]$ and $D[x\langle x := P_i\rangle]$ respectively. The final lines in both sequences then follow and we have $\mathcal{T}(M) \longrightarrow^{+}_{Var} \mathcal{T}(N)$, one reduction for each copy of the redex $C_2[x]\langle x := P\rangle$.

Case $M \equiv C[P\langle x := Q\rangle] \ \overrightarrow{\ }_{\mathrm{D}}\ C[P] \equiv N, \quad x \notin \mathrm{fv}(P)$**.**

Figure 2.23 displays the general term for $\mathcal{T}(M)$ followed by a series of reductions. Figure 2.24 displays the general term for $\mathcal{T}(N)$.

The terms M_1 and $\mathcal{T}(N)$ appear very similar. One difference is that $\mathcal{T}(N)$ has the weakenings W_V corresponding to the variables lost in the Λ_{sub} term at top-level. We first pull the weakenings Π_5 and their copies (for each copy of the Λ_{sub} redex induced by the translation) upwards through the term, merging them to reach M_2.

M_2 now closely resembles $\mathcal{T}(N)$ except for two differences: i) M_2 contains extra weakenings such as W_{Π_3} and W_{Π_4} for each copy of the Λ_{sub} redex and ii) it also contains contractions involving such sets of variables Π_3 and Π_4. We pull all of these weakenings up, merging them with the extra contractions to reach a $\longrightarrow_{Wk}$ normal form M_3. Ignoring the placement of contractions, M_3 is now equivalent to $\mathcal{T}(N)$.

Finally, we can conclude that this term M_3 is equivalent to $\mathcal{T}(N)$: as $\mathcal{T}(M)$ has its contractions at their most efficient, so does M_3 by Corollary 39.

□

Corollary 46 (PSN for Λ_{sub}). *$\overrightarrow{\ }_{\mathrm{ACD}}$ PSN of $\overrightarrow{\ }_{\beta}$.*

Proof.

Case $\Rightarrow$**.** Let M be any pure term which is strongly normalising for $\overrightarrow{\ }_{\beta}$. As M is strongly normalising for $\overrightarrow{\ }_{\beta}$, $\mathcal{T}(M) = (\downarrow_{\mathrm{p}\downarrow\mathrm{c}}(\mathcal{A}(M))$ is strongly normalising for $\longrightarrow_{\lambda\mathrm{blxr}}$ by Conjecture 40. By Proposition 45, any infinite $\overrightarrow{\ }_{\mathrm{ACD}}$ sequence starting from M induces an infinite reduction sequence starting from $\mathcal{T}(M)$. By contrapositive, M is strongly normalising for $\overrightarrow{\ }_{\mathrm{ACD}}$.

Case $\Leftarrow$**.** By Proposition 6.2, infinite $\overrightarrow{\ }_{\beta}$-reductions induce infinite $\overrightarrow{\ }_{\mathrm{ACD}}$-reductions.

□

At the time of writing, this proof strategy (simulating $\overrightarrow{\ }_{\mathrm{ACD}}$ in another explicit substitution calculus) could not work with any other explicit substitution calculus except $\lambda\mathrm{lxr}$ as both PSN and FCS are required.

2.4.5 Sketch of proof of PSN by translation to Λ_I

We have proved PSN by simulating $\overrightarrow{\ }_{\mathrm{ACD}}$-reduction with $\lambda\mathrm{blxr}$ reduction and using the fact that $\lambda\mathrm{blxr}$ has this property. The proof of PSN for $\lambda\mathrm{blxr}$ follows Lengrand's approach of simulating reduction in Λ_I [Len05]. If we combine these simulations, we have a simulation of $\overrightarrow{\ }_{\mathrm{ACD}}$-reduction in Λ_I as in Figure 2.25.

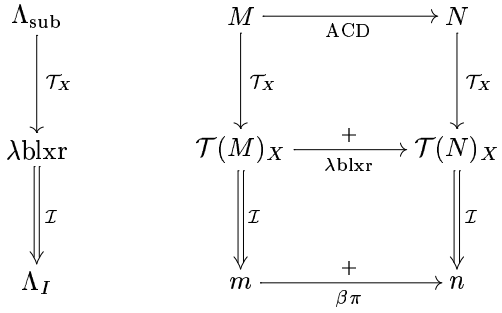


Figure 2.25: Simulation of $\longrightarrow_{\mathrm{ACD}}$-reduction in Λ_I

This immediately suggests that a proof of PSN for Λ_{sub} may be given by using a translation to Λ_I directly. We hope to explore this approach in future work. The composition of $\mathcal{T}$ and $\mathcal{I}$ seems overkill as the translation duplicates substitutions and adds weakenings corresponding to the index set X.

Instead, we initially propose[16] the relation $\mathcal{I}$ defined in Figure 2.26. $\mathcal{I}$ relates Λ_{sub} terms with explicit substitutions to λ_I terms where the substitutions have been finished. The "memory operator" keeps track of garbage substitutions which are otherwise discarded in the relation. This is necessary in Λ_{sub} which does not have the linearity property (which λlxr enjoys) that all binders bind a free occurrence of a variable.

$$\frac{}{x \mathrel{\mathcal{I}} x} \qquad \frac{M \mathrel{\mathcal{I}} m}{\lambda x.M \mathrel{\mathcal{I}} \lambda x.m}\, x \in \mathrm{fv}(M) \qquad \frac{M \mathrel{\mathcal{I}} m}{\lambda x.M \mathrel{\mathcal{I}} \lambda x.[m,x]}\, x \notin \mathrm{fv}(M)$$

$$\frac{M \mathrel{\mathcal{I}} m \qquad N \mathrel{\mathcal{I}} n}{MN \mathrel{\mathcal{I}} mn} \qquad \frac{M \mathrel{\mathcal{I}} m}{M \mathrel{\mathcal{I}} [m,p]}\, p \in \Lambda_I$$

$$\frac{M \mathrel{\mathcal{I}} m \qquad N \mathrel{\mathcal{I}} n}{M\langle x := N\rangle \mathrel{\mathcal{I}} M\{x \setminus n\}}\, x \in \mathrm{fv}(M) \qquad \frac{M \mathrel{\mathcal{I}} m \qquad N \mathrel{\mathcal{I}} n}{M\langle x := N\rangle \mathrel{\mathcal{I}} [M,n]}\, x \notin \mathrm{fv}(M)$$

Figure 2.26: Relating Λ_{sub} terms with λ_I terms

[16] Stéphane Lengrand has since suggested a relation with less rules, replacing the condition $x \in \mathrm{fv}(M)$ in two of the rules with $x \in \mathrm{fv}(m)$ and removing the two rules with the condition $x \notin \mathrm{fv}(M)$.

Chapter 3

Extensions and other ideas

We finish with some ideas which presented themselves during our exploration of Λ_{sub} and $'\Lambda_{\mathrm{BIG}}$.

3.1 Proposed extension to Λ_{sub}

The composition rule of $\lambda\mathrm{xc}^{-}$,

$$M\langle x := P\rangle\langle y := Q\rangle \;\xrightarrow[\mathrm{c}]{}\; M\langle x := P\langle y := Q\rangle\rangle \quad \text{if } x \in \mathrm{fv}(\downarrow_{\mathrm{xgc}}(M)), y \notin \mathrm{fv}(M).$$

does not break PSN. It seems natural to assume that an extension of Λ_{sub} with this rule would also retain PSN. However, it is not clear how to specify the condition $x \in \mathrm{fv}(\downarrow_{\mathrm{CD}}(M))$ in the bigraphical encoding. We therefore propose that Λ_{sub} may be extended with the more general rule (which breaks PSN for $\lambda\mathrm{xgc}$) from $\lambda\mathrm{xc}$:

$$M\langle x := P\rangle\langle y := Q\rangle \;\xrightarrow[\mathrm{c}]{}\; M\langle x := P\langle y := Q\rangle\rangle \quad \text{if } y \notin \mathrm{fv}(M).$$

This rule can be naturally encoded in the bigraphical setting. The condition $y \notin \mathrm{fv}(M)$ is captured by the inner interface of the parametric reaction rule depicted below.

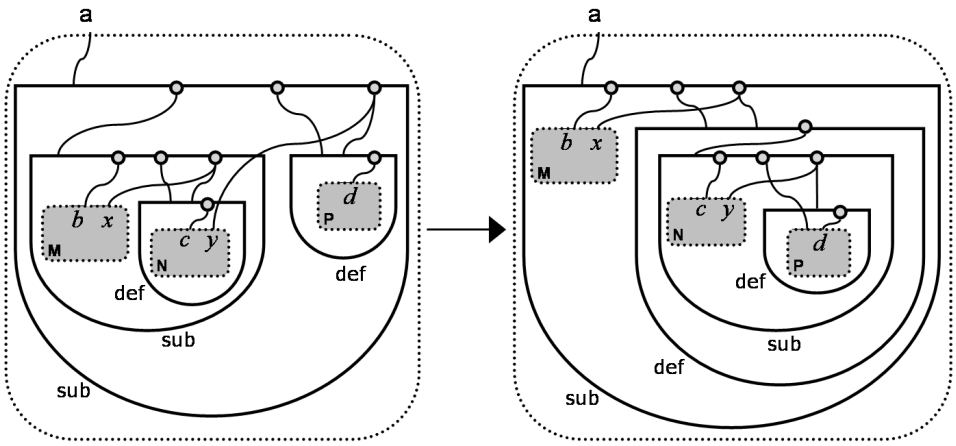


Figure 3.1: An explicit composition rule for $'\Lambda_{\mathrm{BIG}}$.

We hypothesize that the extension is confluent and satisfies PSN but do not attempt a proof here. It does not seem to break PSN as it does not allow any new substitutions to take place – any free y in N can be replaced by P in the original Λ_{sub}. Another reason we believe the extension to be safe is that the composition seems to preserve $\mathrm{bigSN}_{\mathrm{ACD}}$.

However, it would seem that this extension is not very useful in the presence of wide substitution – if $y \notin \mathrm{fv}(M)$ then it would seem that a reduction sequence starting from $M\langle x := P\rangle\langle y := Q\rangle$ could be mimicked by $M\langle x := P\langle y := Q\rangle\rangle$.

3.2 Initial translation to λlxr

Our initial encoding of Λ_{sub} into λlxr did not create mobile and idle copies of substitutions and was considerably simpler:

Definition (Encoding of Λ_{sub} terms in λlxr). *The encoding $\mathcal{L}$ of Λ_{sub} terms in λlxr is defined on pure terms as*

$$
\begin{array}{ll}
\mathcal{L}(x)_{X \uplus x} := W_X(x) & \\
\mathcal{L}(\lambda x.M)_X := W_{X \setminus \mathrm{fv}(M)}(\lambda x.\mathcal{L}(M)) & \text{if } x \in \mathrm{fv}(M) \\
\mathcal{L}(\lambda x.M)_X := W_{X \setminus \mathrm{fv}(M)}(\lambda x.W_x(\mathcal{L}(M)) & \text{if } x \notin \mathrm{fv}(M) \\
\mathcal{L}(MN)_X := W_Y \left(C_\Phi^{\Delta,\Pi}\left(R_\Delta^\Phi(\mathcal{L}(M))\ R_\Pi^\Phi(\mathcal{L}(N))\right)\right) & \\
\mathcal{L}(M\langle x := N\rangle)_X := W_Y\ C_\Theta^{\Psi,\Gamma}\left(\mathcal{L}(R_\Psi^\Theta(M))\langle x := \mathcal{L}(R_\Gamma^\Theta(N))\rangle\right) & \text{if } x \in \mathrm{fv}(M) \\
\mathcal{L}(M\langle x := N\rangle)_X := W_Y\ C_\Theta^{\Psi,\Gamma}\left(W_x(\mathcal{L}(R_\Psi^\Theta(M)))\langle x := \mathcal{L}(R_\Gamma^\Theta(N))\rangle\right) & \text{if } x \notin \mathrm{fv}(M)
\end{array}
$$

where $Y = X \setminus (\mathrm{fv}(M) \cup \mathrm{fv}(N))$, $\Phi := \mathrm{fv}(M) \cap \mathrm{fv}(N)$, $\Theta = (\mathrm{fv}(M) \setminus \{x\}) \cap \mathrm{fv}(N)$.
When a translation is not tagged, X is assumed to be $\emptyset$.

The encoding of explicit substitutions falls directly out of the other encodings and the $\longrightarrow_B$ rule. This encoding does not duplicate substitutions and is easier to reason about. As we now explain, it was insufficient to simulate $\overrightarrow{\mathrm{ACD}}$ reduction in λlxr.

First, this alternate encoding also required altering λlxr to achieve simulation. Take $\mathcal{L}$ as the translation between Λ_{sub} and λlxr and consider the square:

$$
\begin{array}{ccc}
\Lambda_{\mathrm{sub}} & x\langle x := y\rangle & \xrightarrow[\mathrm{C}]{} y\langle x := y\rangle \\
\Big\downarrow \mathcal{L} & \Big\downarrow \mathcal{L} & \Big\downarrow \mathcal{L} \\
\lambda\mathrm{lxr} & x\langle x := y\rangle & \overset{}{\underset{\lambda\mathrm{lxr}}{\dashrightarrow\!\!\!\!\!\!\gg}} C_y^{y_1,y_2}\left((W_x(y_1))\langle x := y_2\rangle\right).
\end{array}
$$

There is no series of λlxr reductions to fill in the dotted arrow. We require a substitution to be performed but a $\longrightarrow_{Var}$ reduction would automatically garbage collect the substitution. Our solution was to break up the $\longrightarrow_{Var}$ reduction so that the substitution was not discarded after reduction. We defined the rule

$$x\langle x := M\rangle \ \longrightarrow_{(Var_{\mathrm{S}})}\ C_\Phi^{\Delta,\Pi}\left(\left(W_x R_\Delta^\Phi(M)\right)\langle x := R_\Pi^\Phi(M)\rangle\right) \quad \text{where } \Phi = \mathrm{fv}(M).$$

A $\longrightarrow_{Vars} \longrightarrow_{Weak1} \longrightarrow^*_{Merge}$ sequence then performs a $\longrightarrow_{Var}$ reduction.

Next, for similar reasons to our current encoding, a translation from Λ_{sub} to λlxr required a normal form (again $(x(xx))\langle x := y\rangle \ \overrightarrow{\mathrm{C}}\ (x(yx))\langle x := y\rangle$ demonstrates this). The first 'normal form' we tried was similar to $\overrightarrow{\mathrm{pc}}$ but it also pulled weakenings out as far as possible (although this may be implicit in the translation). However, the union of $WApp2$ and $App2$ is not confluent. Consider:

$$(v(W_x y)\langle x := p\rangle)_{App2} \longleftarrow (v(W_x y))\langle x := p\rangle \longrightarrow_{WApp2} W_x(vy)\langle x := p\rangle.$$

This lead us to use the current $\overrightarrow{\mathrm{pc}}$ form.

Finally, the problem with this encoding was demonstrated in Section 2.4.1. Take the reduction $x\langle y := z\rangle\langle z := p\rangle \ \overrightarrow{\mathrm{D}}\ x\langle x := p\rangle$. Without using a normal form in the translation, the encoding $(W_y(x)\langle y := z\rangle)\langle z := p\rangle$ can reduce to $(W_z x)\langle z := p\rangle$ but when using the normal form, we find that $W_y(x)\langle y := z\langle z := p\rangle\rangle$ cannot reduce to $(W_z x)\langle z := p\rangle$. So if we use the normal form, we can simulate $\overrightarrow{\mathrm{C}}$ but not $\overrightarrow{\mathrm{D}}$ whilst if we simply use the encoding, the reverse is true. This problem is depicted in Figure 3.2, where $\longrightarrow_{Wk}$ is the union of the rules for pulling weakenings upwards and merging them with contractions. This problem lead us to the idea of using mobile and idle copies of substitutions in the encoding.

This initial encoding was simpler yet insufficient. However, it may be possible to formulate an alternative translation using this encoding where substitutions are only duplicated prior to pushing inside garbage (to simulate the $\overrightarrow{\mathrm{D}}$ rule properly). This would involve splitting the $(Comp)$ reduction as below.

$$
\begin{array}{lll}
M\langle x := P\rangle\langle y := Q\rangle & \longrightarrow_{Comp_1} & M\langle x := P\langle y := Q\rangle\rangle \\
 & & \text{where } y \in \mathrm{fv}(P), x \in \mathrm{fv}(M) \\
M\langle x := P\rangle\langle y := Q\rangle & \longrightarrow_{Comp_2} & C_\Phi^{\Delta,\Pi}\left(W_{y'}(M\langle x := P\langle y := Q_1\rangle\rangle)\langle y' := Q_2\rangle\right) \\
 & & \text{where } y \in \mathrm{fv}(P), x \notin \mathrm{fv}(M)
\end{array}
$$

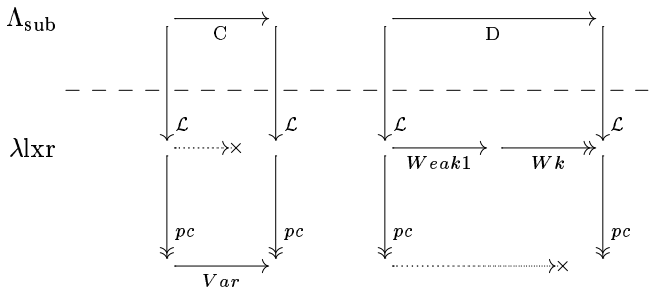


Figure 3.2: Problems with initial translation and normal form

We would also require the (Var_{S}) rule. PSN needs to be proved for this new calculus but it seems likely. λblxr has PSN and it duplicates substitutions in much the same way, by creating garbage copies of existing substitutions which cannot propagate through the term. This translation may yield a simpler proof than the current one as less duplication of substitutions would occur.

3.3 Alternative encodings of the λ-calculus

This was work explored with Martin Elsman who had an idea for an alternative encoding of the λ-calculus using explicit environments. The aims were to have an encoding with a notion similar to pointer/reference passing where all free occurrences of a variable could be replaced in one reaction step. To this end, we tried encoding variables as links. An encoding of explicit environments was never realised however.

Among the reasons why the encoding we did try (without explicit environments) did not work was that (i) reduction could destroy occurrences of terms and (ii) with this encoding, it was not possible to give an inductive encoding of λ-terms via composition – in general, this should probably be a bad indication when encoding a calculus where the terms are defined inductively.

Another attempt (by the author) at an alternative encoding was by using a flattened structure *i.e.* all controls were atomic and nesting was encoding via linking. Variables were represented as controls again. This had many of the same ideas as $'\Lambda$BIG including explicit substitution and garbage collection. As controls were atomic, this implied that terms would be built from ground up via prime products and fusions of wiring – again this might have been a warning sign. The problem here was to do with the binding of names – on closer inspection, we are almost redefining $'\Lambda$BIG without the benefits of being able to employ binding ports. While sorting the links may have solved this problem, it is unclear whether anything would have been gained by this approach. In fact, we were completely ignoring the place graph structure in the bigraphs which naturally lends itself to calculi with an inductive definition for terms.

3.4 A new property of controls

During early explorations of $'\Lambda$BIG, we considered attempting to add a new property to controls in bigraphical signatures. We named this property *exclusive* with the idea that in a bigraph with exclusive controls, reaction may only occur under these controls. We were motivated by the idea of restricting reaction in $'\Lambda$BIG so that all reduction sequences were of the form $\xrightarrow[\mathrm{CD}]{}\!\!\!\!\!\rightarrow\ \xrightarrow[\mathrm{A}]{}\ \xrightarrow[\mathrm{CD}]{}\!\!\!\!\!\rightarrow\ \xrightarrow[\mathrm{A}]{}\ \xrightarrow[\mathrm{CD}]{}\!\!\!\!\!\rightarrow\ \xrightarrow[\mathrm{A}]{}\ \cdots$. This would give an immediate correspondence between $'\Lambda$BIG and the λ-calculus.

The questions of whether this idea adds anything new to bigraph theory, whether it could be encoded using existing means (*e.g.* as a reaction rule covering controls and sites with passive controls) or whether it would require extensions to the theory, or whether there are any suitable applications have not been addressed.

However, besides these questions, there is also the question of whether this restriction would be beneficial for $'\Lambda$BIG. This type of restriction was applied to $\lambda\sigma$ by Goubault-Larrecq [GL96] to prove that simply-typed terms were strongly normalising under this reduction subrelation. However, Kesner and Lengrand point out that this restriction does not benefit from the possibilities that explicit substitution calculi offer – to delay substitutions or to partially apply them [KL05]. Another argument against this restriction is that one could also imagine implementations where terms were distributed over different processors and reduction was performed concurrently.

Chapter 4

Summary

4.1 Conclusions and related work

The first aim of this report was to prove that $'\Lambda_{\mathrm{BIG}}$ is confluent. Milner proved weak confluence in [Mil05b] and left open the challenge to prove strong confluence. Although our proof is not bigraphical and does not advance the understanding of confluence in bigraphical systems, it does show that $'\Lambda_{\mathrm{BIG}}$ has this desirable property on closed terms (terms without metavariables).

The question of whether $'\Lambda_{\mathrm{BIG}}$ or Λ_{sub} has open confluence – confluence on terms with metavariables – remains unanswered and would be more interesting to the bigraphical community than closed confluence, which only represents confluence on ground terms. In related work in explicit substitutions, David and Guillaume [DG01] state that composition of substitutions is necessary for open confluence. They point out that λ_d [Kes96] and $\lambda\sigma_n$ [Rit99] have weak composition of substitutions which is not enough to get open confluence.

It is possible that much of our proof of closed confluence could be rewritten in $'\Lambda_{\mathrm{BIG}}$ using Milner's proofs for deciding weak confluence. For example, in Propositions 1, the proofs of diamond property and local confluence could be proved using Milner's theorems in [Mil05b] whereas the proofs of strong normalisation might be done by labelling the bigraph terms. The proofs of propositions involving free names (*e.g.* Proposition 5) should follow from the translation from Λ_{sub} to $'\Lambda_{\mathrm{BIG}}$. The proof of confluence is based on the three main properties; (1) confluence of the original calculus, (2) that the substitution calculus is a conservative extension, and (3) the generalised interpretation method [KR97]. Bloo and Rose's approach may be useful for proving closed confluence of bigraphical encodings with explicit substitution of other calculi besides the λ-calculus.

Our proof of confluence was achieved by identifying the strong connection between Λ_{sub} and $\lambda\mathrm{xgc}$. Proving PSN – our other main aim – using their inductive proofs has proven trickier. The proof for $\Lambda_{\mathrm{subC}^\flat}$ was straightforward but composition of substitutions makes the inductive proof of PSN for Λ_{sub} much more involved. The intersection of $\mathrm{bigSN}_{\mathrm{ACD}}$ and $\#\mathrm{gf}$ identify the subset of strongly normalising terms of Λ_{sub} but do not yield a simple characterisation. Intersection types [CDC78, CDC80] may provide such a characterisation. Lengrand et al. [LLD$^+$04] have successfully used intersection types to characterise the strongly normalising terms of $\lambda\mathrm{xgc}$.

We identified various subsets of $\Lambda\mathrm{x}$ which are relevant to the different reduction relations we have studied. Figure 4.1 combines many of our examples and proofs by depicting how these subsets are related, where the solid arrows denote subset inclusion. The proofs of the implications are denoted in the figure: all are either proof by definition, contrapositive, or example. Each subset excludes some $\Lambda\mathrm{x}$ terms which are not strongly normalising for $\overrightarrow{\mathrm{ACD}}$ or β-reduction. From the diagram, we can see that the addition of levels of inter-substitution reduction (composition of substitutions) decreases the set of strongly normalising terms. We also restate our hypothesis that if we disallow any inter-substitution reduction (*i.e.* $\Lambda_{\mathrm{sub}\mathbb{C}}$) then the resulting calculus is extremely similar to $\lambda\mathrm{xgc}$. This may be investigated in future work but we think that the proofs should follow from Section 2.2.2.

We explored the relationship between Λ_{sub}, $'\Lambda_{\mathrm{BIG}}$, and $\lambda\mathrm{lxr}$ and introduced a modified (and less efficient) calculus $\lambda\mathrm{blxr}$ based on the $\lambda\mathrm{lxr}$ calculus of Kesner and Lengrand. We then gave a proof of PSN for Λ_{sub} by translating Λ_{sub} terms into $\lambda\mathrm{blxr}$ and simulating $\overrightarrow{\mathrm{ACD}}$ reduction using the translation. Our novel idea was the duplication of substitutions (mobile and idle copies). The proof of simulation is somewhat involved but we feel that it is simpler than the inductive method, which relied on 'regressing' terms. As we have remarked, this form of proof could not have been achieved with any other calculus as $\lambda\mathrm{lxr}$ is the first published calculus with PSN and FCS. It may also be possible to prove PSN for Λ_{sub} directly by relating terms of Λ_{sub} terms with terms of λ_I and

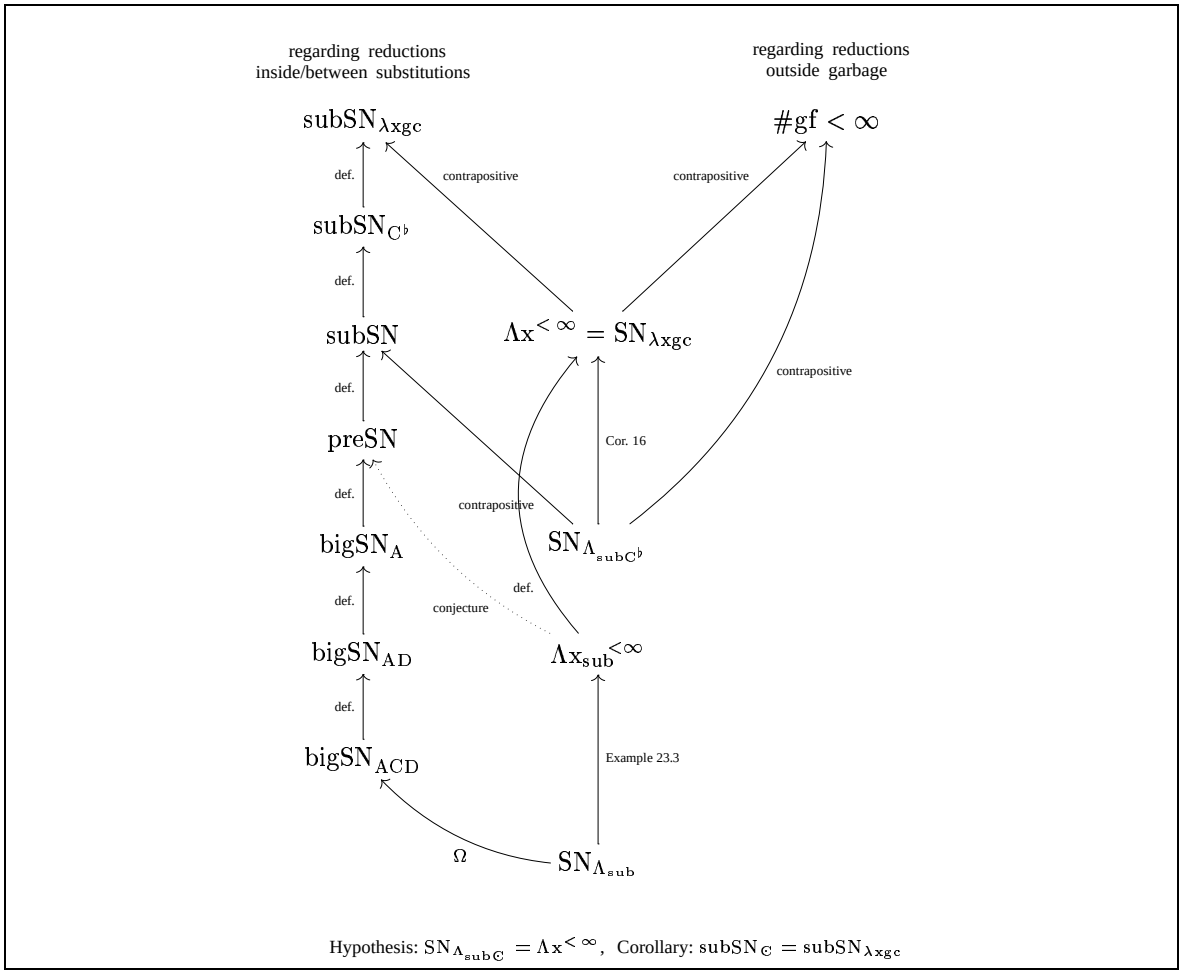


Figure 4.1: Relationship between properties of Λx terms

then applying Lengrand's techniques [Len05].

In this report, we study how the reduction relation of Λ_{sub} matches that of the λ-calculus. We are concerned here about properties such as confluence and termination in the presence of explicit substitution. From a process calculus perspective, Bundgaard and Hildebrandt [BH05] have encoded the *Higher-Order Mobile Embedded Resources* (*Homer*) calculus as a Brs with explicit substitution and garbage collection. They base their presentation on $'\Lambda_{\mathrm{BIG}}$ and in [ibid.], they prove an operational correspondence between Homer and their encoding.

There seems to be some overlap between the theories of interaction nets, the ρ-calculus, and bigraphs. Some overlaps are by design – for example, interaction nets inspired aspects of bigraph theory [JM04] – but others may be interesting to explore and we briefly mention them now.

In recent work [FMS06], Fernández, Mackie, and Sinot introduced an encoding of the ρ-calculus in bigraphical nets. Bigraphical nets add the notion of locality (via a nesting structure *i.e.* place graph) to interaction nets. The encoding takes advantage of wide reaction.

Kesner and Lengrand use λlxr to demonstrate the connection between higher-order substitution and proofs in linear logic proofs by translating λlxr into proof-nets [Gir87]. The anonymous referees of our HOR 2006 submission based on Section 2.4 suggested that we should explore the relationship between Λ_{sub} and proof-nets.

The notion of wide/external substitution appears in the cyclic λ-calculus of Ariola and Klop [AK97]. The ρ_{g}-calculus of Bertolissi et al. [Ber05, BBCK05] extends the ρ-calculus to handle graph-like structures with cycles and sharing and is a natural extension of the cyclic λ-calculus of Ariola and Klop [AK97]. The ρ_{g}-calculus allows non-local or *external* substitution as do bigraphs and is confluent under linearity conditions [Ber05]. Again, it remains an open problem to find sufficicent conditions for confluence in general bigraphical systems.

In conclusion, we claim that $'\Lambda$BIG is a very natural encoding of the λ-calculus, with step-by-step simulation of β-reduction and closed confluence. Besides λlxr, it is the only explicit substitution calculus to date which also preserves strong normalisation of β-reduction and enjoys full composition of substitutions.

4.2 Acknowledgements

The work arose from work with Martin Elsman and Thomas Hildebrandt during a stay, kindly facilitated by Lars Birkedal and Annette Hjort Knudsen, with the Bigraphical Programming Languages group at the IT University of Copenhagen.

Many thanks are due to Martin Elsman and Thomas Hildebrandt from whom I was offered much help and many remarks on this document. The visit was also made possible by the support of my supervisor, Mícheál Mac an Airchinnigh. This research and that visit was partially supported by funding from the Irish Research Council for Science, Engineering and Technology: funded by the National Development Plan.

Many thanks are due to Stéphane Lengrand for taking time to step through some of my technical details with me. I also wish to thank Kristoffer Rose and François-Régis Sinot for their helpful correspondence and references, Malcolm Dowse for many helpful conversations about this work, and the anonymous referees of my submission to the 3rd International Workshop on Higher-Order Rewriting (HOR 2006) for their useful comments and for pointing out a technical error.

Bibliography

[ACCL91] Martín Abadi, Luca Cardelli, Pierre-Louis Curien, and Jean-Jacques Lévy. Explicit substitutions. *Journal of Functional Programming*, 1(4):375–416, 1991.

[AK97] Zena M. Ariola and Jan Willem Klop. Lambda calculus with explicit recursion. *Information and Computation*, 139(2):154–233, 1997.

[Bar84] H. P. Barendregt. *The Lambda Calculus: Its Syntax and Semantics*, volume 103 of *Studies in Logic and the Foundations of Mathematics*. North-Holland, revised edition, 1984.

[BBCK05] Clara Bertolissi, Paolo Baldan, Horatiu Cirstea, and Claude Kirchner. A rewriting calculus for cyclic higher-order term graphs. *Electronic Notes in Theoretical Computer Science*, 127(5):21–41, 2005.

[Ber05] Clara Bertolissi. The graph rewriting calculus : confluence and expressiveness. In Mario Coppo, Elena Lodi, and G. Michele Pinna, editors, *9th Italian conference on Italian Conference on Theoretical Computer Science (ICTCS 2005), Siena, Italy*, volume 3701 of *Lecture Notes in Computer Science*, pages 113–127. Springer Verlag, Oct 2005.

[BG99] Roel Bloo and Herman Geuvers. Explicit substitution: on the edge of strong normalization. *Theoretical Computer Science*, 211(1–2):375–395, 1999.

[BH05] Mikkel Bundgaard and Thomas Hildebrandt. Bigraphical semantics of higher-order mobile embedded resources with local names. In Reiko Heckel, Barbara König, and Arend Rensink, editors, *Proceedings of GT-VC '05 (Graph Transformation for Verification and Concurrency)*, number 05–34 in CTIT Technical Reports. Centre for Telematics and Information Technology, University of Twente, 2005.

[BKdV00] Inge Bethke, Jan Willem Klop, and Roel C. de Vrijer. Descendants and origins in term rewriting. *Information and Computation*, 159(1-2):59–124, 2000.

[Blo97] Roel Bloo. *Preservation of termination for explicit substitution*. PhD thesis, Eindhoven University of Technology, 1997.

[BR95] Roel Bloo and Kristoffer Høgsbro Rose. Preservation of strong normalisation in named lambda calculi with explicit substitution and garbage collection. In *CSN-95: Computer Science in the Netherlands*, November 1995.

[CDC78] Mario Coppo and Mariangiola Dezani-Ciancaglini. A new type assignment for λ-terms. *Archiv f ur mathematische Logik und Grundlagenforschung*, 19:139–156, 1978.

[CDC80] Mario Coppo and Mariangiola Dezani-Ciancaglini. An extension of the basic functionality theory for the λ-calculus. *Notre Dame Journal of Formal Logic*, 21(4):685–693, October 1980.

[DG01] René David and Bruno Guillaume. A lambda-calculus with explicit weakening and explicit substitution. *Mathematical Structures in Computer Science*, 11(1):169–206, 2001.

[FKP96] Maria C. F. Ferreira, Delia Kesner, and Laurence Puel. Lambda-calculi with explicit substitutions and composition which preserve beta-strong normalization. In Michael Hanus and Mario Rodríguez-Artalejo, editors, *ALP*, volume 1139 of *Lecture Notes in Computer Science*, pages 284–298. Springer, 1996.

[FM99] Maribel Fernández and Ian Mackie. Closed reductions in the lambda-calculus. In Jörg Flum and Mario Rodríguez-Artalejo, editors, *Computer Science Logic*, volume 1683 of *Lecture Notes in Computer Science*, pages 220–234. Springer, 1999.

[FMS06] Maribel Fernández, Ian Mackie, and François-Régis Sinot. Interaction nets vs. the ρ-calculus: Introducing bigraphical nets. *Electronic Notes in Theoretical Computer Science*, 154(3):19–32, 2006.

[Gir87] Jean-Yves Girard. Linear logic. *Theoretical Computer Science*, 50:1–102, 1987.

[GL96] Jean Goubault-Larrecq. A proof of weak termination of typed lambda-sigma-calculi. In Eduardo Giménez and Christine Paulin-Mohring, editors, *TYPES*, volume 1512 of *Lecture Notes in Computer Science*, pages 134–153. Springer, 1996.

[Har89] Thérèse Hardin. Confluence results for the pure strong categorical logic CCL: λ-calculi as subsystems of CCL. *Theoretical Computer Science*, 65(3):291–342, 1989.

[Hue80] Gérard Huet. Confluent reductions: Abstract properties and applications to term rewriting systems. *Journal of the ACM*, 27(4):797–821, 1980.

[JM04] Ole Høgh Jensen and Robin Milner. Bigraphs and mobile processes (revised). Technical Report UCAM-CL-TR-580, Computer Laboratory, University of Cambridge, February 2004.

[Kes96] Delia Kesner. Confluence properties of extensional and non-extensional λ-calculi with explicit substitutions (extended abstract). In Harald Ganzinger, editor, *RTA*, volume 1103 of *Lecture Notes in Computer Science*, pages 184–199. Springer, 1996.

[KL] Delia Kesner and Stéphane Lengrand. Explicit operators for lambda-calculus. Available at "*http://www.pps.jussieu.fr/∼kesner/papers/*".

[KL05] Delia Kesner and Stéphane Lengrand. Extending the explicit substitution paradigm. In Jürgen Giesl, editor, *RTA*, volume 3467 of *Lecture Notes in Computer Science*, pages 407–422. Springer, 2005.

[Klo95] Jan Willem Klop. Term graph rewriting. In Gilles Dowek, Jan Heering, Karl Meinke, and Bernhard Möller, editors, *Higher-Order Algebra, Logic, and Term Rewriting*, volume 1074 of *Lecture Notes in Computer Science*, pages 1–16. Springer, 1995.

[KR97] Fairouz Kamareddine and Alejandro Ríos. Extending a λ-calculus with explicit substitution which preserves strong normalisation into a confluent calculus on open terms. *Journal of Functional Programming*, 7(4), July 1997.

[Laf90] Yves Lafont. Interaction nets. In *POPL '90: Proceedings of the 17th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 95–108, New York, NY, USA, 1990. ACM Press.

[Lei01] James J. Leifer. *Operational congruences for reactive systems*. PhD thesis, Computer Laboratory, University of Cambridge, 2001. Available in revised form as Technical Report 521, Computer Laboratory, University of Cambridge, 2001.

[Len05] Stéphane Lengrand. Induction principles as the foundation of the theory of normalisation: Concepts and techniques. Technical report, PPS laboratory, Université Paris 7, March 2005. available at `http://hal.ccsd.cnrs.fr/ccsd-00004358`.

[LLD+04] Stéphane Lengrand, Pierre Lescanne, Dan Dougherty, Mariangiola Dezani-Ciancaglini, and Steffen van Bakel. Intersection types for explicit substitutions. *Information and Computation*, 189(1):17–42, 2004.

[LM04] James J. Leifer and Robin Milner. Transition systems, link graphs and petri nets. Technical Report UCAM-CL-TR-598, Computer Laboratory, University of Cambridge, August 2004.

[Mel95] Paul-André Melliès. Typed lambda-calculi with explicit substitutions may not terminate. In *Proceedings of the Second International Conference on Typed Lambda Calculi and Applications, Edinburgh*, number 902 in Lecture Notes in Computer Science, pages 328–334, 1995.

[Mil90] Robin Milner. Functions as processes. Technical Report RR-1154, INRIA Sophia-Antipolis, February 1990.

[Mil01] Robin Milner. Bigraphical reactive systems: basic theory. Technical report, Computer Laboratory, University of Cambridge, 2001.

[Mil04] Robin Milner. Local bigraphs, confluence and λ-calculus (draft), 2004.

[Mil05a] Robin Milner. Bigraphs: A tutorial, 2005.

[Mil05b] Robin Milner. Pure bigraphs. Technical Report UCAM-CL-TR-614, Computer Laboratory, University of Cambridge, 2005.

[Muñ96] César Muñoz. Confluence and preservation of strong normalisation in an explicit substitutions calculus (extended abstract). In *Proc. Eleventh Annual IEEE Symposium on Logic in Computer Science*, pages 440–447, New Brunswick, New Jersey, July 1996. IEEE Computer Society Press.

[New42] M.H.A. Newman. On theories with a combinatorial definition of 'equivalence'. *Annals of Mathematics*, 43(2):223–243, 1942.

[O'C06] Shane O'Conchúir. Proving PSN after ruining a perfectly good calculus. Technical Report TCD-CS-2006-49, Trinity College Dublin, September 2006.

[PV98] Joachim Parrow and Björn Victor. The fusion calculus: Expressiveness and symmetry in mobile processes. In *Proceedings of LICS '98*. IEEE, Computer Society Press, June 1998.

[Rit99] Eike Ritter. Characterising explicit substitutions which preserve termination. In Jean-Yves Girard, editor, *TLCA*, volume 1581 of *Lecture Notes in Computer Science*, pages 325–339. Springer, 1999.

[Ros92] Kristoffer Høgsbro Rose. Explicit cyclic substitutions. In M. Rusinowitch and J.-L. Rémy, editors, *CTRS '92–3rd International Workshop on Conditional Term Rewriting Systems*, number 656 in Lecture Notes in Computer Science, pages 36–50, Pont-a-Mousson, France, 1992. Springer-Verlag.

[Ros96a] Kristoffer Høgsbro Rose. Explicit substitution – tutorial & survey, 1996.

[Ros96b] Kristoffer Høgsbro Rose. *Operational Reduction Models for Functional Programming Languages*. PhD thesis, DIKU, University of Copenhagen, Denmark, Universitetsparken 1, DK-2100 København Ø, February 1996. Available as DIKU report 96/1.

[Ter03] Terese. *Term Rewriting Systems*, volume 55 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 2003.

Appendix A

Appendices

A.1 Lemmas for inductive proof of PSN for Λ_{sub}

Lemma 47 ($\xrightarrow[\mathrm{AD}]{}$ SN).

Proof. $\xrightarrow[\mathrm{A}]{}$ reductions preserve the number of variable occurrences in a term whereas $\xrightarrow[\mathrm{D}]{}$ reductions decrease this number. The proof follows from the finiteness of terms and $\xrightarrow[\mathrm{A}]{}$-SN. □

Lemmas 48 (preservation, reflection of preSN).

1. *If* $\mathrm{preSN}(M)$ *and* $M \xrightarrow[\mathrm{AD}]{} N$ *is inside substitution then* $\mathrm{preSN}(N)$.
2. *If* $\mathrm{preSN}(M)$ *and* $M \xrightarrow[\mathrm{D}]{} N$ *is outside substitution then* $\mathrm{preSN}(N)$.
3. *If* $\mathrm{preSN}(M)$ *and* $M \xrightarrow[\mathrm{C}]{} N$ *then* $\mathrm{preSN}(N)$.
4. *If* $\mathrm{preSN}(N)$ *and* $M \xrightarrow[\mathrm{A}]{} N$ *is inside substitution then* $\mathrm{preSN}(M)$.

Proof.

1. Let $R_j \xrightarrow[\mathrm{AD}]{} R'_j$ be the reduction inside some substitution $\langle y_j := R_j \rangle$. We must show that
   (a) $R'_j\langle y_{j+1} \dots y_n\rangle$ is strongly normalising. As $\mathrm{preSN}(M)$, $R_j\langle y_{j+1} \dots y_n\rangle$ is strongly normalising. As $R_j\langle y_{j+1} \dots y_n\rangle \xrightarrow[\mathrm{AD}]{} R'_j\langle y_{j+1} \dots y_n\rangle$, the case is proved.
   (b) for any body of substitution R_i below $\langle y_j := R_j\rangle$,
   $R_i\langle y_{i+1} \dots y_{j-1}\rangle\langle y_j := R'_j\rangle\langle y_{j+1} \dots y_n\rangle$ is strongly normalising.
   Again, as $\mathrm{preSN}(M)$, $R_i\langle y_{i+1} \dots y_{j-1}\rangle\langle y_j := R_j\rangle\langle y_{j+1} \dots y_n\rangle$ is strongly normalising and $R_i\langle y_{i+1} \dots y_{j-1}\rangle\langle y_j := R_j\rangle\langle y_{j+1} \dots y_n\rangle \xrightarrow[\mathrm{AD}]{} R_i\langle y_{i+1} \dots y_{j-1}\rangle\langle y_j := R'_j\rangle\langle y_{j+1} \dots y_n\rangle$.
2. Let $\langle x := Q\rangle$ be the discarded substitution. For any body of substitution P in N, P is either outside of, above, or below $\langle x := Q\rangle$ in M. In the first two cases, $P\langle y_1 \dots y_n\rangle$ is strongly normalising in N by $\mathrm{preSN}(M)$. In the third case, let $\langle x := Q\rangle$ lie below $\langle y_i := R_i\rangle$. By $\mathrm{preSN}(M)$, $P\langle y_1 \dots y_{i-1}\rangle\langle x := Q\rangle\langle y_i \dots y_n\rangle$ is strongly normalising. Hence, $P\langle y_1 \dots y_n\rangle$ is strongly normalising.
3. Let $C[x]\langle x := T\rangle$ be the $\xrightarrow[\mathrm{C}]{}$ redex.
   If x lies inside a body of substitution, $\mathrm{preSN}(N)$ by $\mathrm{preSN}(M)$.
   Otherwise, if T does not contain any substitutions then $\mathrm{preSN}(N)$ again by $\mathrm{preSN}(M)$. Say T contains substitutions $\langle z_1 := Q_1\rangle \dots \langle z_m := Q_m\rangle$. Let $\langle y_1 := R_1\rangle, \dots, \langle y_{i-1} := R_{i-1}\rangle$ be the substitutions above x in C. We must prove $Q_1\langle z_2 \dots z_m\rangle\langle y_1 \dots y_{i-1}\rangle\langle x := T\rangle\langle y_i \dots y_n\rangle$ is strongly normalising. Variable capture does not occur and so $\{y_1, \dots, y_{i-1}, x\}$ are not free names of $Q_1\langle z_2 \dots z_m\rangle$. It remains to show that:
   - $Q_1\langle z_2 \dots z_m\rangle\langle y_i \dots y_n\rangle$, and
   - $R_1\langle y_2 \dots y_{i-1}\rangle\langle x := T\rangle\langle y_i \dots y_n\rangle$

(where $\langle y_1 \dots y_n \rangle$ are the substitutions above $\langle x := T \rangle$ in M and N) are strongly normalising. These follow from $\mathrm{preSN}(M)$.

4. Assume that $\mathrm{preSN}(M)$ is false. Then there exists a body of substitution P in M such that $P\langle y_1 \dots y_n \rangle$ is not strongly normalising. There are two cases.

   (i) $P \xrightarrow[\mathrm{A}]{} P'$ and $Q \equiv P'\langle y_1 \dots y_n \rangle$ is strongly normalising or

   (ii) $R_i \xrightarrow[\mathrm{A}]{} R_i'$ and $Q \equiv P\langle y_1 \dots y_{i-1} \rangle \langle y_i := R_i' \rangle \langle y_{i+1} \dots y_n \rangle$ is strongly normalising

   In either case, the graph below shows that any infinite reduction sequence starting from $P\langle y_1 \dots y_n \rangle$ can be matched by a reduction sequence from Q.

$$\begin{array}{ccccccc} P\langle y_1 \dots y_n \rangle & \xrightarrow{\mathrm{ACD}} & \cdot & \xrightarrow{\mathrm{ACD}} & \cdot & \xrightarrow{\mathrm{AC}} & \cdots \\ {\scriptstyle \mathrm{A}}\downarrow & & {\scriptstyle \mathrm{A}}\downarrow & & {\scriptstyle \mathrm{A}}\downarrow & & \\ Q & \xrightarrow{\mathrm{ACD}} & \cdot & \xrightarrow{\mathrm{ACD}} & \cdot & \xrightarrow{\mathrm{ACD}} & \cdots \end{array}$$

   Each square can be filled as follows. If the top reduction is $\xrightarrow[\mathrm{A}]{}$, then either the square is filled by $\xrightarrow[\mathrm{A}]{} \Diamond$, or else the top and left reductions are the same in which case the bottom line has joined the top.

   If the top reduction is $\xrightarrow[\mathrm{C}]{}$ then there are three subcases identified by Milner which are filled in as in Lemma 49.2.

   If the top reduction is $\xrightarrow[\mathrm{D}]{}$ then there are three subcases.

   - The redexes are independent. we have one-step confluence [Mil04].
   - The $\xrightarrow[\mathrm{A}]{}$ redex lies beneath the $\xrightarrow[\mathrm{D}]{}$ redex. It is either discarded and the top line joins the bottom, or else one vertical $\xrightarrow[\mathrm{A}]{}$ reduction completes the square.
   - The $\xrightarrow[\mathrm{D}]{}$ redex lies beneath the $\xrightarrow[\mathrm{A}]{}$ redex. One vertical $\xrightarrow[\mathrm{A}]{}$ reduction completes the square.

   Hence, $\mathrm{preSN}(N)$ is false which is a contradiction. Thus, $\mathrm{preSN}(M)$.

□

Definition ($\xrightarrow[\mathrm{C}]{}$ reduction creates new $\xrightarrow[\mathrm{A}]{}$ redexes). *A reduction $M \xrightarrow[\mathrm{C}]{} M'$ is said to* create new $\xrightarrow[\mathrm{A}]{}$ redexes *when*

$$M \equiv C[(NQ)]\langle x := P \rangle \xrightarrow[\mathrm{C}]{} C[(N'Q)]\langle x := P \rangle \equiv M'$$

where $\downarrow_{\mathrm{D}}(N) \equiv x$, $\downarrow_{\mathrm{D}}(P) \equiv (\lambda y.P')$, and the reduction replaces a free x in N which perseveres as $\downarrow_{\mathrm{D}}(N)$. The same is said of any such reduction which occurs under some context.

These $\xrightarrow[\mathrm{C}]{}$ reductions do not preserve $\mathrm{bigSN}_{\mathrm{AD}}$ in general (see Example 52). An alternative description would be: for a reduction $M \xrightarrow[\mathrm{C}]{} M'$ which creates a new $\xrightarrow[\mathrm{A}]{}$ redex, there is a path $M' \xrightarrow[\mathrm{D}]{\gg} M''$ such that there is a $\xrightarrow[\mathrm{A}]{}$ redex in M'' which is not a residual of any $\xrightarrow[\mathrm{A}]{}$ redex in M. The general form of these reductions (omitting outer contexts) is:

$$\begin{array}{rl} & \big((x\langle u_1 \dots u_n \rangle)(M)\big)\langle x := (\lambda y.N)\langle v_1 \dots v_m \rangle \rangle \\ \xrightarrow[\mathrm{C}]{} & \Big(\big((\lambda y.N)\langle v_1 \dots v_m \rangle \langle u_1 \dots u_n \rangle\big)\big(M\big)\Big)\langle x := (\lambda y.N)\langle v_1 \dots v_m \rangle \rangle, \end{array}$$

where $x \neq u_i, 1 \leq i \leq n, v_j \notin \mathrm{fv}(\lambda y.N), 1 \leq j \leq m$. The last term can then reduce:

$$\xrightarrow[\mathrm{D}]{\gg} \quad \big((\lambda y.N')(M')\big)\langle x := (\lambda y.N)\langle v_1 \dots v_m \rangle \rangle;$$

(where $N \xrightarrow[\mathrm{D}]{\gg} N'$ and $M \xrightarrow[\mathrm{D}]{\gg} M'$) and contains a new $\xrightarrow[\mathrm{A}]{}$ redex.

The normal-forms in the definition account for the cases when the new ' $\xrightarrow[\mathrm{A}]{}$ redex to be' is blocked by garbage. When this garbage is collected, the $\xrightarrow[\mathrm{A}]{}$ redex is then enabled to fire as above.

In the following proofs, we use the following definition of $\mathrm{bigSN}_{\mathrm{AD}}$.

Definition ($\mathrm{bigSN}_{\mathrm{AD}}$). *The predicate $\mathrm{bigSN}_{\mathrm{AD}}(M)$ states that for all sequences $M \xrightarrow[\mathrm{AD}]{\gg} M_i$ containing only reductions outside substitution, $\mathrm{preSN}(M_i)$.*

Lemmas 49 (preservation of $\mathrm{bigSN}_{\mathrm{AD}}$).

1. *If* $\mathrm{bigSN}_{\mathrm{AD}}(M)$ *and* $M \xrightarrow[\mathrm{AD}]{} M_1$ *then* $\mathrm{bigSN}_{\mathrm{AD}}(M_1)$.
2. *If* $\mathrm{bigSN}_{\mathrm{AD}}(M)$ *and* $M \xrightarrow[\mathrm{C}]{} M_1$ *does not create any new* $\xrightarrow[\mathrm{A}]{}$ *redexes outside substitution then* $\mathrm{bigSN}_{\mathrm{AD}}(M_1)$.

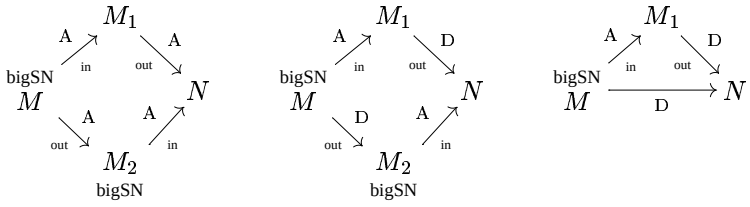
Proof.

1. We induct over $\mathrm{maxred}_{\mathrm{AD}}(M)$. The base case is when $n = 1$. We need to prove that $\mathrm{preSN}(M_1)$. If the reduction is outside substitution then the follows by definition. If it is inside substitution then it follows by Lemma 48.1.

   In the inductive case, if $M \xrightarrow[\mathrm{AD}]{} M_1$ is a reduction outside substitution then the proof follows by definition. Let $M \xrightarrow[\mathrm{AD}]{} M_1$ be a reduction inside substitution and $M_1 \xrightarrow[\mathrm{AD}]{} N$ a reduction outside substitution. $\mathrm{preSN}(M_1)$ by Lemma 48.1. Proving $\mathrm{bigSN}_{\mathrm{AD}}(N)$ will complete the proof.

   The redex of $M_1 \xrightarrow[\mathrm{AD}]{} N$ is a residual of an redex outside substitution in M as $M \xrightarrow[\mathrm{AD}]{} M_1$ cannot move the redex outside substitution. There is then a reduction $M \xrightarrow[\mathrm{AD}]{} M_2$ outside substitution corresponding to $M_1 \xrightarrow[\mathrm{AD}]{} N$.

   When $M \xrightarrow[\mathrm{A}]{} M_1$, there are three cases (see [Mil04]) depicted below:

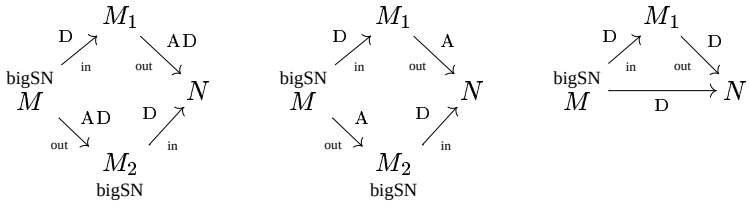


   (a) $M_1 \xrightarrow[\mathrm{A}]{} N$. Then $M_2 \xrightarrow[\mathrm{A}]{} N$ by $\xrightarrow[\mathrm{A}]{} \Diamond$. As $M \xrightarrow[\mathrm{A}]{} M_2$ is outside substitution, $\mathrm{bigSN}_{\mathrm{AD}}(M_2)$ by definition. By the induction hypothesis, $\mathrm{bigSN}_{\mathrm{AD}}(N)$.

   (b) $M_1 \xrightarrow[\mathrm{D}]{} N$ and the redexes $M \xrightarrow[\mathrm{A}]{} M_1$ and $M \xrightarrow[\mathrm{D}]{} M_2$ are independent. The proof then follows the last case.

   (c) $M_1 \xrightarrow[\mathrm{D}]{} N$ and the reactum of $M \xrightarrow[\mathrm{A}]{} M_1$ lies inside the substitution of the redex of $M_1 \xrightarrow[\mathrm{D}]{} N$ (it is discarded). By definition, $\mathrm{bigSN}_{\mathrm{AD}}(N)$.

   When $M \xrightarrow[\mathrm{D}]{} M_1$, there are three cases (see [Mil04]) depicted below:



   (a) $M_1 \xrightarrow[\mathrm{AD}]{} N$ with the redexes of $M \xrightarrow[\mathrm{D}]{} M_1$ and $M \xrightarrow[\mathrm{AD}]{} M_2$ independent. By [Mil04], we have one-step confluence. As $M \xrightarrow[\mathrm{A}]{} M_2$ is outside substitution, $\mathrm{bigSN}_{\mathrm{AD}}(M_2)$ by definition. By the induction hypothesis, $\mathrm{bigSN}_{\mathrm{AD}}(N)$.

   (b) $M_1 \xrightarrow[\mathrm{A}]{} N$ and the redex of $M \xrightarrow[\mathrm{D}]{} M_1$ lies under the redex of $M_1 \xrightarrow[\mathrm{A}]{} N$. It should be clear that the reduction graph is as depicted. The proof then follows the last case.

   (c) $M_1 \xrightarrow[\mathrm{D}]{} N$ and the reactum of $M \xrightarrow[\mathrm{D}]{} M_1$ lies under the redex of $M_1 \xrightarrow[\mathrm{D}]{} N$ (it is discarded). By definition, $\mathrm{bigSN}_{\mathrm{AD}}(N)$.

2. $\mathrm{preSN}(M_1)$ by Lemma 48.3. For any reduction $M_1 \xrightarrow[\mathrm{AD}]{} N$ outside substitution, proving $\mathrm{bigSN}_{\mathrm{AD}}(N)$ will complete the proof.

   Our general strategy is to join the sequences $M \xrightarrow[\mathrm{C}]{} M_1 \xrightarrow[\mathrm{AD}]{} N$ and $M \xrightarrow[\mathrm{AD}]{} M_2$ (where the $\xrightarrow[\mathrm{AD}]{}$ redex in M_1 is a residual of the redex in M) knowing that $\mathrm{maxred}_{\mathrm{AD}}(M_2) < \mathrm{maxred}_{\mathrm{AD}}(M)$ and $\mathrm{bigSN}_{\mathrm{AD}}(M_2)$ by 1 above. We induct over $\mathrm{maxred}_{\mathrm{AD}}$.

   > If $\mathrm{maxred}_{\mathrm{AD}}(M) = n$, $\mathrm{bigSN}_{\mathrm{AD}}(M)$, and $M \xrightarrow[\mathrm{C}]{} M_1$ does not create any new $\xrightarrow[\mathrm{A}]{}$ redexes outside substitution then $\mathrm{bigSN}_{\mathrm{AD}}(M_1)$. (I.H.)

   For the base case, $n = 0$ and as M_1 has no $\xrightarrow[\mathrm{AD}]{}$ redexes outside substitution, $\mathrm{preSN}(M_1)$ then implies $\mathrm{bigSN}_{\mathrm{AD}}(M_1)$.

   As each $\xrightarrow[\mathrm{AD}]{}$ redex outside substitution in M_1 is a residual of a redex in M, we break the inductive case down over the ways in which the $\xrightarrow[\mathrm{AD}]{}$ and $\xrightarrow[\mathrm{C}]{}$ redexes in M can overlap. These subcases were identified by Milner in the bigraphical setting of $'\Lambda_{\mathrm{BIG}}$ [Mil04].

   Let $C[x]\langle x := R\rangle$ be the $\xrightarrow[\mathrm{C}]{}$ redex and $(\lambda y.P)Q$ or $P\langle y := Q\rangle$ be the $\xrightarrow[\mathrm{AD}]{}$ redex. We will omit some Λ_{sub} contexts in the cases below for clarity.

   case 1 Independent redexes. Theorem 4.3 [Mil04] proves that there is a reduction graph as in Figure A.1(a). $\mathrm{maxred}_{\mathrm{AD}}(M_2) < \mathrm{maxred}_{\mathrm{AD}}(M)$ and $\mathrm{bigSN}_{\mathrm{AD}}(M_2)$. As $M \xrightarrow[\mathrm{C}]{} M_1$ does not create any new $\xrightarrow[\mathrm{A}]{}$ redexes, neither does $M_2 \xrightarrow[\mathrm{C}]{} N$. Thus, $\mathrm{bigSN}_{\mathrm{AD}}(N)$ by the induction hypothesis.

   case 2 One redex lies beneath the other. There are two possibilities.

   - The $\xrightarrow[\mathrm{A}]{}$ redex is a subterm of R in the $\xrightarrow[\mathrm{C}]{}$ redex. We then have a reduction graph as in Figure A.1(b).
     $\mathrm{maxred}_{\mathrm{A}}(M_2) < \mathrm{maxred}_{\mathrm{A}}(M)$ and $\mathrm{bigSN}_{\mathrm{AD}}(M_2)$. As $M \xrightarrow[\mathrm{C}]{} M_1$ does not create any new $\xrightarrow[\mathrm{A}]{}$ redexes outside substitution (both displayed redexes are residuals) neither does $M_2 \xrightarrow[\mathrm{C}]{} M_3$. Thus, $\mathrm{bigSN}_{\mathrm{AD}}(M_3)$ by the induction hypothesis. By Lemma 48.4, $\mathrm{preSN}(N)$. To complete the proof, we need to show that if the path $N \equiv N_1 \xrightarrow[\mathrm{AD}]{} N_m$ contains only reductions outside substitution then $\mathrm{preSN}(N_m)$. Our strategy is to join up the paths in the diagram below where $N_m \xrightarrow[\mathrm{A}]{} M_{2+m}$ is the residual of the redex $N_1 \xrightarrow[\mathrm{A}]{} M_3$.

     $$\begin{array}{ccc} N_1 & \xrightarrow[\mathrm{AD}]{n} & N_m \\ {\scriptstyle \mathrm{A}}\downarrow & & {\scriptstyle 0:1}\downarrow{\scriptstyle \mathrm{A}} \\ M_3 & \xrightarrow[\mathrm{AD}]{n} & M_{2+m} \end{array}$$

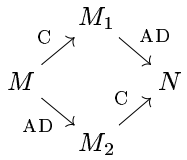
     Each step in the top path may be joined to the bottom path as one of the following three cases.

     $$\begin{array}{ccc} N_i & \xrightarrow[\mathrm{A}]{} & N_{i+1} \\ {\scriptstyle \mathrm{A}}\downarrow & & \downarrow{\scriptstyle \mathrm{A}} \\ M_{2+i} & \xrightarrow[\mathrm{A}]{} & M_{3+i} \end{array} \qquad \begin{array}{ccc} N_i & \xrightarrow[\mathrm{D}]{} & N_{i+1} \\ {\scriptstyle \mathrm{A}}\downarrow & & \downarrow{\scriptstyle \mathrm{A}} \\ M_{2+i} & \xrightarrow[\mathrm{D}]{} & M_{3+i} \end{array} \qquad \begin{array}{ccc} N_i & \xrightarrow[\mathrm{D}]{} & N_{i+1} \\ {\scriptstyle \mathrm{A}}\downarrow & & \Vert \\ M_{2+i} & \xrightarrow[\mathrm{D}]{} & M_{3+i} \end{array}$$

     The first follows by $\xrightarrow[\mathrm{A}]{}\Diamond$. The second case is when the $\xrightarrow[\mathrm{A}]{}$ and $\xrightarrow[\mathrm{D}]{}$ redexes are independent. In the third case, the $\xrightarrow[\mathrm{D}]{}$ redex discards the $\xrightarrow[\mathrm{A}]{}$ substitution.
     As $\mathrm{bigSN}_{\mathrm{AD}}(M_3), \mathrm{bigSN}_{\mathrm{AD}}(M_{2+m})$. By Lemma 48.4, $\mathrm{preSN}(N_m)$.
   - The $\xrightarrow[\mathrm{C}]{}$ redex lies beneath the $\xrightarrow[\mathrm{A}]{}$ redex either as a subterm of P or Q. We prove the former subcase – the latter is similar.
     Let $P \equiv C'\big[C[x]\langle x := R\rangle\big]$. We have a reduction graph as in Figure A.1(c). $\mathrm{maxred}_{\mathrm{A}}(M_2) < \mathrm{maxred}_{\mathrm{A}}(M)$ and $\mathrm{bigSN}_{\mathrm{AD}}(M_2)$. As $M \xrightarrow[\mathrm{C}]{} M_1$ does not create any new $\xrightarrow[\mathrm{A}]{}$ redexes, neither does $M_2 \xrightarrow[\mathrm{C}]{} N$. Thus, $\mathrm{bigSN}_{\mathrm{AD}}(N)$ by the induction hypothesis.

   case 3 The variable x of the $\xrightarrow[\mathrm{C}]{}$ redex lies beneath the $\xrightarrow[\mathrm{A}]{}$ redex. The substitution definition $x := R$ lies outside.

   Let x lie beneath P so $P \equiv C[x]$. We have a reduction graph as in Figure A.1(d). The remainder of the proof follows the previous case and the proof for x beneath Q is similar.



(a) Case 1

$$
\begin{array}{ccc}
C[x]\langle x := \overset{M}{C'[(\lambda y.P)Q]}\rangle & \xrightarrow{\text{C}} & C\big[C'[(\lambda y.P)Q]\big]\langle x := \overset{M_1}{C'[(\lambda y.P)Q]}\rangle \\
\Big\downarrow{\scriptstyle \text{A}} & & \Big\downarrow{\scriptstyle \text{A}} \\
C[x]\langle x := \overset{M_2}{C'[P\langle y := Q\rangle]}\rangle & & C\big[C'[P\langle y := Q\rangle]\big]\langle x := \overset{N}{C'[(\lambda y.P)Q]}\rangle \\
\Big\downarrow{\scriptstyle \text{C}} & \swarrow{\scriptstyle \text{A}} & \\
C\big[C'[P\langle y := Q\rangle]\big]\langle x := \overset{M_3}{C'[P\langle y := Q\rangle]}\rangle & &
\end{array}
$$

(b) Case 2 (i)

$$
\begin{array}{ccc}
\big(\lambda y.C'\big[C[x]\langle x := \overset{M}{R}\rangle\big]\big)Q & \xrightarrow{\text{C}} & \big(\lambda y.C'\big[C[R]\langle x := \overset{M_1}{R}\rangle\big]\big)Q \\
\Big\downarrow{\scriptstyle \text{A}} & & \Big\downarrow{\scriptstyle \text{A}} \\
C'\big[C[x]\langle x := \overset{M_2}{R}\rangle\big]\langle y := Q\rangle & \xrightarrow[\text{C}]{} & C'\big[C[R]\langle x := \overset{N}{R}\rangle\big]\langle y := Q\rangle
\end{array}
$$

(c) Case 2 (ii)

$$
\begin{array}{ccc}
C'\big[(\lambda y.C[x])Q\big]\langle x := \overset{M}{R}\rangle & \xrightarrow{\text{C}} & C'\big[(\lambda y.C[R])Q\big]\langle x := \overset{M_1}{R}\rangle \\
\Big\downarrow{\scriptstyle \text{A}} & & \Big\downarrow{\scriptstyle \text{A}} \\
C'\big[C[x]\langle y := \overset{M_2}{Q}\rangle\big]\langle x := R\rangle & \xrightarrow{\text{C}} & C'\big[C[R]\langle y := \overset{N}{Q}\rangle\big]\langle x := R\rangle
\end{array}
$$

(d) Case 3

Figure A.1: Reduction graphs for Lemma 49.2.

case 4 This case cannot occur between a $\xrightarrow[\text{C}]{}$ and a $\xrightarrow[\text{AD}]{}$ redex (see [Mil04]).

case 5 This case cannot occur between a $\xrightarrow[\text{C}]{}$ and a $\xrightarrow[\text{AD}]{}$ redex (see [Mil04]).

□

A.2 Properties of reduction in Λ_{sub}

This section contains examples of properties in Λ_{sub} relevant to the proof of PSN.

Example 50 ($\xrightarrow[\mathrm{D}]{}$ is necessary for some infinite sequences.). *All variables below are distinct.*

$$\begin{array}{rl} & \big((\lambda x.\lambda y.yy)z\big)\,(\lambda v.vv) \\ \xrightarrow[\mathrm{A}]{} & \big((\lambda y.yy)\langle x := z\rangle\big)\,(\lambda v.vv) \\ \xrightarrow[\mathrm{D}]{} & (\lambda y.yy)(\lambda v.vv) \end{array}$$

Example 51 ($\xrightarrow[\mathrm{D}]{}$ does not preserve $\mathrm{bigSN}_{\mathrm{A}}$.). *All variables below are distinct.*

$$\begin{array}{rl} & \Big(\big(\lambda y.(\lambda x.z)\big)\,w\Big)\,\Big(\Omega\Big) \\ \xrightarrow[\mathrm{A}]{} & \big((\lambda x.z)\langle y := w\rangle\big)\,(\Omega) \equiv M \\ \xrightarrow[\mathrm{D}]{} & \big((\lambda x.z)\big)\,(\Omega) \equiv M' \\ \xrightarrow[\mathrm{A}]{} & z\langle x := \Omega\rangle \end{array}$$

$\mathrm{preSN}(M)$ *and since* M *has no* $\xrightarrow[\mathrm{A}]{}$ *redexes,* $\mathrm{bigSN}_{\mathrm{A}}(M)$. *However, the* $\xrightarrow[\mathrm{D}]{}$ *reduction creates a* $\xrightarrow[\mathrm{A}]{}$ *redex outside substitution in* M' *such that* $\mathrm{bigSN}_{\mathrm{A}}(M')$ *is false.*

Example 52 ($\xrightarrow[\mathrm{C}]{}$ does not preserve $\mathrm{bigSN}_{\mathrm{AD}}$.). *Consider the sequence*

$$\begin{array}{rcl} M & \equiv & \big((x)(\Omega)\big)\langle x := \lambda w.y\rangle \\ & \xrightarrow[\mathrm{C}]{} & \big((\lambda w'.y)(\Omega)\big)\langle x := \lambda w.y\rangle \\ & \xrightarrow[\mathrm{A}]{} & y\langle w' := \Omega\rangle\langle x := \lambda w.y\rangle. \end{array}$$

with distinct variables. $\mathrm{bigSN}_{\mathrm{AD}}(M)$, *but the* $\xrightarrow[\mathrm{C}]{}$ *reduct of* M *does not satisfy* $\mathrm{bigSN}_{\mathrm{AD}}$. *What happens here is that a subterm of* M *which is not strongly normalising is moved inside a substitution by a combination of a* $\xrightarrow[\mathrm{C}]{}$ *creating a* $\xrightarrow[\mathrm{A}]{}$ *redex followed by the firing of the latter redex. If we replace* $\lambda w.y$ *with* $(\lambda w.y)\langle u := P\rangle$, $u \neq y$, *then the* $\xrightarrow[\mathrm{C}]{}$ *reduction creates an outer* $\xrightarrow[\mathrm{D}]{}$ *redex leading to a similar situation. These situations are analogous to the sequence*

$$\begin{array}{rcl} M & \equiv & (\lambda x.M)(\Omega) \\ & \xrightarrow[\mathrm{b}]{} & M\langle x := \Omega\rangle. \end{array}$$

in $\lambda\mathrm{xgc}$ *which shows that* subSN *is not generally preserved by* $\xrightarrow[\mathrm{bxgc}]{}$.

A.3 Interleaving β-reductions in Λ_{sub}

Take the term $(\lambda x.(\lambda y.xy)z)w$ with two β-redexes. It has the reduction graph in Figure A.2.

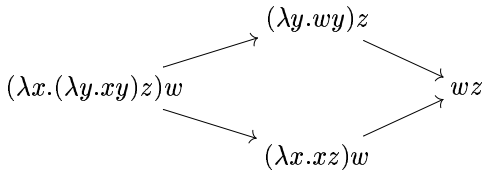


Figure A.2: Reduction graph of $(\lambda x.(\lambda y.xy)z)w$ in the λ-calculus

In Λ_{sub}, a β-reduction corresponds to a sequence of reductions. These reductions may interleave as in Figure A.3. In the figure, the outer reduction paths correspond to the firing of one β-reduction after the other. The terms on the inside of the graph are terms where both β-reductions have been partially completed.

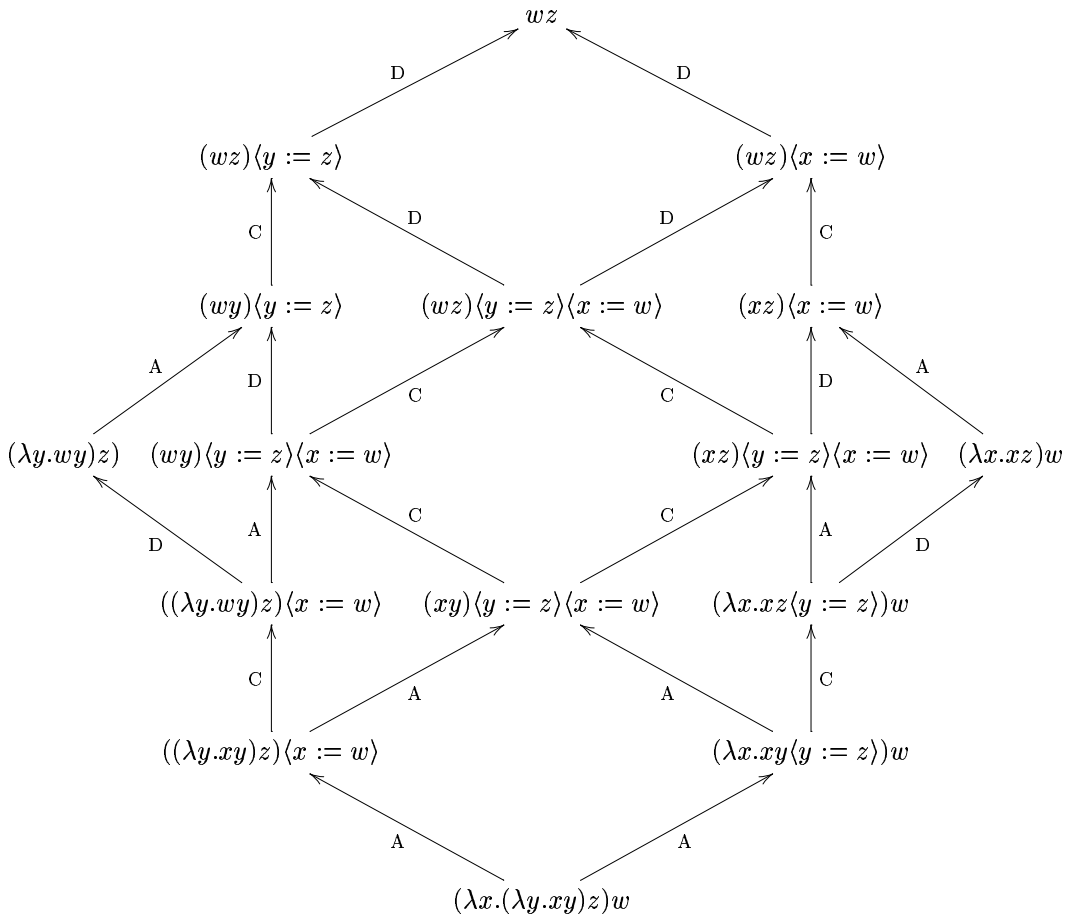


Figure A.3: Reduction graph of $(\lambda x.(\lambda y.xy)z)w$ in Λ_{sub}

A.4 Contraction graphs

λlxr has three explicit constructors – explicit substitutions, weakenings, and contractions. The first is familiar and the second is uncomplicated. Contractions are not complicated but we found that a graphical representation aided our initial reasoning by providing some visual proofs. In this section, we present this representation of the contractions of a λlxr term.

Contractions are used to provide linearity when multiple occurrences of a free variable x exist in a term by renaming the occurrences and then 'aliasing' them to x. For example, the λ-term xx is encoded in λlxr as $\mathcal{A}(xx) \equiv C_x^{x_1,x_2}(x_1x_2)$. In general, $n-1$ contractions are needed to 'linearise' n free occurrences of some variable. For example,

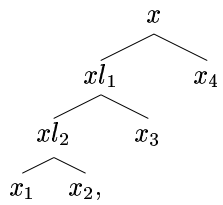
$$\mathcal{A}(((xx)x)x) = C_x^{xl_1,x_4}\big(C_{xl_1}^{xl_2,x_3}(C_{xl_2}^{x_1,x_2}(x_1,x_2)x_3)x_4\big),$$
$$\mathcal{A}((xx)(xx)) = C_x^{xl_1,xl_2}\big(C_{xl_1}^{x_1,x_2}(x_1x_2)C_{xl_2}^{x_3,x_4}(x_3x_4)\big).$$

We have adopted some conventions in the examples above which we typically use in this document. We label the contractions in such a way that the variables which stand for x (which we call *aliases* here) are indexed numerically from innermost to outermost, left to right. There are special labels which tie the contractions together which we label as xl indexed with some integer. We adopt these conventions for our graphical representation below.

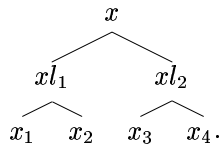
A contraction $C_x^{x_1,x_2}$ may be represented as a tree

with the 'real' variable on top, the first alias below to the left, and the second alias below to the right. In any term M, there is an obvious nesting structure to contractions given by the abstract syntax tree of M. For example, in the encoding of $((xx)x)x$ above, $C_x^{xl_1,x_4}$ lies above $C_{xl_1}^{xl_2,x_3}$. $C_x^{xl_1,x_4}$ is also connected in some way to $C_{xl_1}^{xl_2,x_3}$ – xl_1 is a free name of the second contraction which is bound in the first contraction. We can therefore directly represent the contraction substructure of a term graphically using forests of binary trees where the root node of a tree represents an 'actual' variable name x, the leaf nodes represent variables in M, and the other nodes link these variables ultimately to x. We call these forest representations *contraction graphs*.

For example, the contraction graph of $\mathcal{A}(((xx)x)x)$ may be visualised as



and the contraction graph of $\mathcal{A}((xx)(xx))$ as

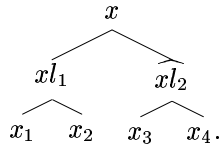


We call the nodes which are neither leaves nor the root *links*.

We find it useful to depict when contractions lie directly beneath each other. For example, in the term

$$\mathcal{A}((y(xx))(xx)) = C_x^{xl_1,xl_2}\big((yC_{xl_1}^{x_1,x_2}(x_1x_2))C_{xl_2}^{x_3,x_4}(x_3x_4)\big),$$

$C_{xl_2}^{x_3,x_4}$ lies directly under $C_x^{xl_1,xl_2}$ whereas $C_{xl_1}^{x_1,x_2}$ does not. When a contraction lies directly under its parent contraction in a term, we indicate this in the graph by decorating the free variable of the inside contraction with a hat ($\widehat{\ }$) so that the contraction graph of this example is



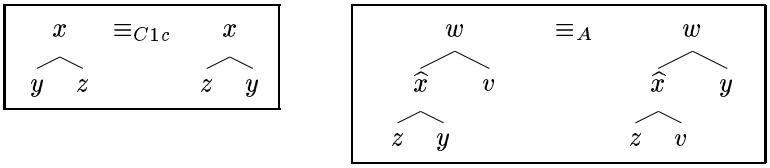


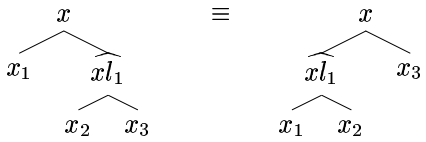
Figure A.4: Graphical representations of $\equiv_{C1c}$ and $\equiv_A$

Using this representation, we can depict the congruences $\equiv_{C1c}$ and $\equiv_A$ as in Figure A.4. The congruence $\equiv_{C1c}$ is visualised by swapping the branches below x. The congruence $\equiv_A$ is visualised by swapping the labels of leaf nodes one generation apart (when the parent of the younger nodes is wearing its hat). The congruence $\equiv_{C2c}$ involves contractions in different trees of the forest and so is not explicitly represented in a contraction graph.

These representations of the congruences may allow easily understood visual proofs involving $\equiv_A$ and $\equiv_{C1c}$. For example, the proof of the equivalence

$$\begin{array}{rl} & C_x^{x_1,xl_1} C_{xl_1}^{x_2,x_3}(x_1(x_2x_3)) \\ \equiv_{C1c} & C_x^{xl_1,x_1} C_{xl_1}^{x_2,x_3}(x_1(x_2x_3)) \\ \equiv_A & C_x^{xl_1,x_3} C_{xl_1}^{x_2,x_1}(x_1(x_2x_3)) \\ \equiv_{C1c} & C_x^{xl_1,x_3} C_{xl_1}^{x_1,x_2}(x_1(x_2x_3)) \end{array}$$

can be read off the diagram below.



Finally, we can extend this representation to depict how a substitution propagates down through contractions in a term via the $\longrightarrow_{Cont1}$ rule as in Figure A.5, where the first three right hand side graphs correspond to the left hand side graphs after a $\longrightarrow_{Cont1}$ reduction and the last two graphs represent a 'tidying up' of the contraction graph.

Using this representation, the contraction graphs of the reduction sequence

$$\begin{array}{rll} & C_x^{x_1,xl_1}\big(x_1 C_{xl_1}^{x_2,x3}(x_2x_3)\big)\langle x := N\rangle & (1) \\ \longrightarrow_{Cont1} & C_\Phi^{\Phi_1,\Phi l_1}\big(\big(x_1 C_{xl_1}^{x_2,x3}(x_2x_3)\big)\langle x_1 := N_1\rangle\langle xl_1 := Nl_1\rangle\big) & \\ \longrightarrow_{App1} & C_\Phi^{\Phi_1,\Phi l_1}\big(\big(x_1\langle x_1 := N_1\rangle C_{xl_1}^{x_2,x3}(x_2x_3)\big)\langle xl_1 := Nl_1\rangle\big) & \\ \longrightarrow_{App2} & C_\Phi^{\Phi_1,\Phi l_1}\big(x_1\langle x_1 := N_1\rangle C_{xl_1}^{x_2,x3}(x_2x_3)\langle xl_1 := Nl_1\rangle\big) & (2) \\ \longrightarrow_{Cont1} & C_\Phi^{\Phi_1,\Phi l_1}\big(x_1\langle x_1 := N_1\rangle C_{\Phi l_1}^{\Phi_2,\Phi_3}\big((x_2x_3)\langle x_2 := N_2\rangle\langle x_3 := N_3\rangle\big)\big) & (3) \\ \longrightarrow^* & C_\Phi^{\Phi_1,\Phi l_1}\big(x_1\langle x_1 := N_1\rangle C_{\Phi l_1}^{\Phi_2,\Phi_3}(x_2\langle x_2 := N_2\rangle x_3\langle x_3 := N_3\rangle)\big) & \end{array}$$

are given in Figure A.6.

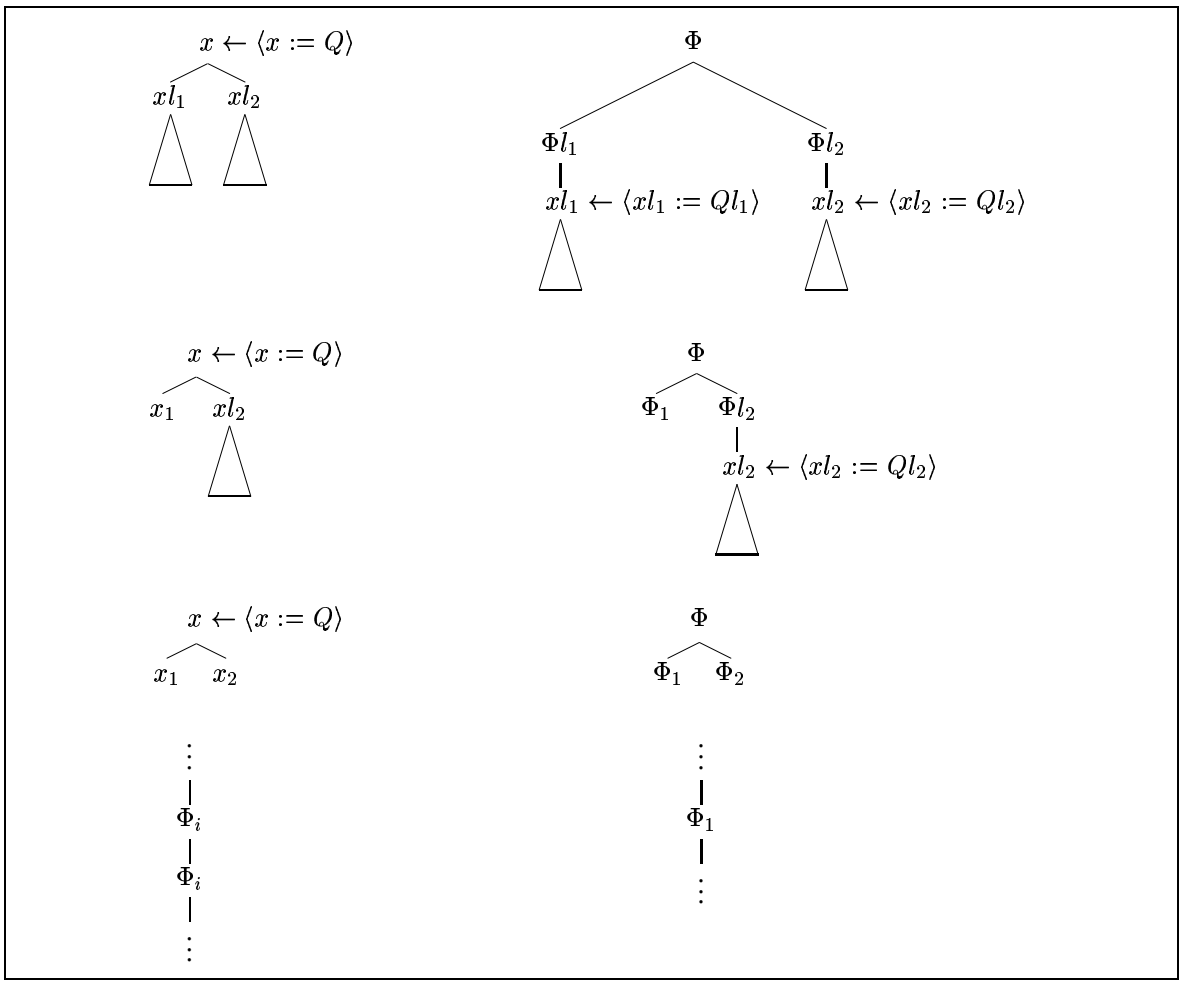
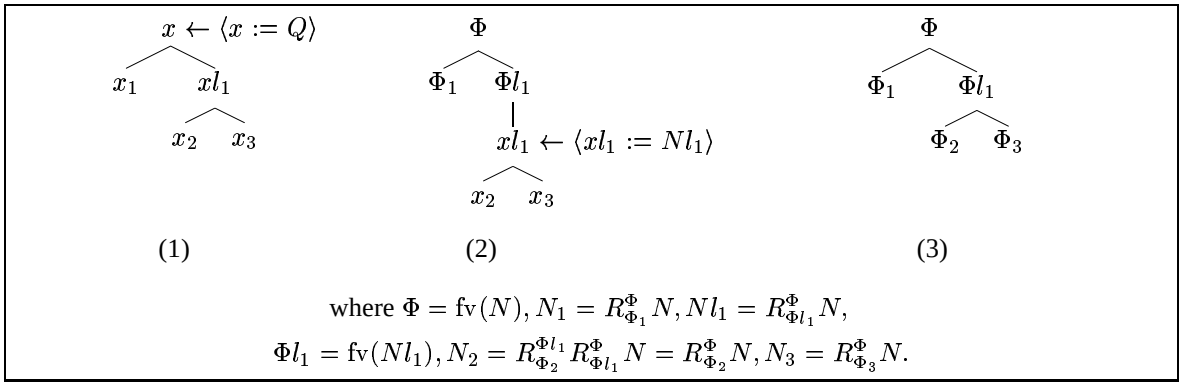


Figure A.5: Substitutions propagating through contraction graphs



where $\Phi = \mathrm{fv}(N), N_1 = R^{\Phi}_{\Phi_1} N, Nl_1 = R^{\Phi}_{\Phi l_1} N,$
$\Phi l_1 = \mathrm{fv}(Nl_1), N_2 = R^{\Phi l_1}_{\Phi_2} R^{\Phi}_{\Phi l_1} N = R^{\Phi}_{\Phi_2} N, N_3 = R^{\Phi}_{\Phi_3} N.$

Figure A.6: A substitution distributing through some contractions