Proving Strong Normalisation via Non-deterministic Translations into Klop's Extended λ-Calculus

Kentaro Kikuchi

RIEC, Tohoku University,
Katahira 2-1-1, Aoba-ku, Sendai 980-8577, Japan
kentaro@nue.riec.tohoku.ac.jp

Abstract

In this paper we present strong normalisation proofs using a technique of non-deterministic translations into Klop's extended λ-calculus. We first illustrate the technique by showing strong normalisation of a typed calculus that corresponds to natural deduction with general elimination rules. Then we study its explicit substitution version, the type-free calculus of which does not satisfy PSN with respect to reduction of the original calculus; nevertheless it is shown that typed terms are strongly normalising with respect to reduction of the explicit substitution calculus. In the same framework we prove strong normalisation of Sørensen and Urzyczyn's cut-elimination system in intuitionistic sequent calculus.

1998 ACM Subject Classification F.4.1 Mathematical Logic

Keywords and phrases Strong normalisation, Klop's extended λ-calculus, Explicit substitution, Cut-elimination

Digital Object Identifier 10.4230/LIPIcs.CSL.2013.395

1 Introduction

It is common to prove strong normalisation of a reduction system by a mapping into a set equipped with a well-founded order, e.g. $(\mathbb{N}, >)$. In the field of λ-calculus, it is also common to use a translation from terms of a calculus into λ-terms that are known to be strongly normalising, e.g. simply typed λ-terms. Such a translation is usually a (deterministic) function, and sometimes gives rise to difficulty in preserving a reduction step of the original calculus in one or more reduction steps of λ-calculus, in particular when the translation involves substitution.

In [19, 20], Lengrand developed a technique to cope with this sort of problem, where the translation from terms of the original calculus is not into λ-terms but into $\lambda I_{[,]}$-terms of [18] with additional pairing constructs. Moreover, it is defined to be non-deterministic (i.e. to be a relation rather than a function) so that an arbitrary term can be added as the second element of the pairing constructs inserted at random places. One can thus retain those terms which would disappear if translated by a function, and preserve reduction steps that take place within those terms. (For a survey on different techniques concerning $\lambda I_{[,]}$ to infer normalisation properties, see, e.g. [8].)

In this paper we first illustrate the technique by proving strong normalisation of typed terms of a calculus that corresponds to natural deduction with general elimination rules [26]. Although the same result has already been shown by different methods (e.g. [12, 22, 24]), our proof will help the reader to understand the contents of the later part of the paper with results that have not been obtained by those methods.

Computer Science Logic 2013 (CSL'13).
Editor: Simona Ronchi Della Rocca; pp. 395–414

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

In the latter half of the paper, we apply the technique to systems with explicit substitutions [1]. We study a modification of the explicit substitution calculus introduced by Nakazawa [22], for which he mentioned the difficulty in proving strong normalisation of typed terms. We explain why the method in [22] does not work for the modified system, and prove strong normalisation of typed terms using a non-deterministic translation into $\lambda I_{[,]}$-calculus. The proof method provides a general framework for showing strong normalisation of systems with various reduction rules on the same terms, which include proof terms for intuitionistic sequent calculus. We illustrate the framework with an extension of Sørensen and Urzyczyn's cut-elimination system [27].

Since Melliès [21] gave an unexpected counter-example, strong normalisation for explicit substitution calculi has been widely studied. For composition-free systems, the methods in [6, 4, 5] are standard. They work even for type-free calculi to prove the Preservation of Strong Normalisation (PSN) property, which states that if a term is strongly normalising in the original calculus without explicit substitutions then it is also strongly normalising in the explicit substitution calculus. This property, however, does not hold for the calculi we treat in this paper. So we use techniques from [19, 20] to prove strong normalisation of typed terms in the explicit substitution calculus, without relying on the result of the original calculus. A similar proof can be found in [16] for the restricted case of proof terms for intuitionistic sequent calculus. In this paper, the definition of the non-deterministic translation is extended and improved from the one in [16]. In [27], a method closely related to the one in [16] has been developed. However, it introduces Klop's pairing constructs not only for λ-terms but also for proof terms for sequent calculus, which leads to complications.

In this paper we will refer to the modification of the system in [22] as $\lambda \mathsf{x}_g$, which makes substitution of the original calculus λ_g explicit in the style of $\lambda \mathsf{x}$ [6]. As mentioned above, the calculus $\lambda \mathsf{x}_g$ does not satisfy PSN with respect to λ_g, but it does not mean a flaw of $\lambda \mathsf{x}_g$. To put it briefly, the reason is that λ_g only implements some specific strategies. (For more details, see Remark in Subsection 3.2.)

The main subject of the paper is the technique of non-deterministic translations into $\lambda I_{[,]}$-calculus. The technique was originally developed for proving PSN of an explicit substitution calculus with composition [14], and later applied to a local cut-elimination procedure that simulates β-reduction [16]. However, the proofs for those systems are not so accessible to readers who are working in other fields. In this paper we explain the key ideas of the technique, separating them from the formalism of explicit substitution calculi. This amounts to extending the range of application of the technique, e.g. to proof of strong normalisation for $\lambda\mu$-calculus [25], solving the so-called erasing continuation problem [23]. (cf. [17])

The paper is organised as follows. In Section 2 we recall the definitions of λ_g-calculus and $\lambda I_{[,]}$-calculus, and prove strong normalisation of typed λ_g-terms. In Section 3 we extend the syntax of λ_g-calculus by explicit substitution, and prove strong normalisation of typed terms by extending the method in Section 2. In Section 4 we apply the proof method to Sørensen and Urzyczyn's cut-elimination system.

2 Strong normalisation for λ_g-calculus

This section provides a survey of the method of proving strong normalisation through a non-deterministic translation into Klop's $\lambda I_{[,]}$-calculus. Although the original formalisation by Lengrand was explained using an explicit substitution calculus with composition [14] (which is the only example to which the technique is applied in [19, 20]), here we apply the method to simply typed λ_g-calculus without explicit substitution.

2.1 λ_g-calculus

λ_g-calculus is introduced as a term calculus corresponding to natural deduction with general elimination rules [26]. It has been studied, e.g. in [11, 22]. Typed terms of the calculus can also be seen as term representation of proofs in a fragment of intuitionistic sequent calculus. We first define the syntax of the type-free version of the calculus.

▶ **Definition 1** (Grammar of λ_g)**.** The set Λ_g of terms of the λ_g-calculus is defined by the following grammar:

$$M, N, P ::= x \mid \lambda x.M \mid M[N, x.P]$$

An element of Λ_g is called a λ_g*-term*. The notions of free and bound variables are defined as usual, with an additional clause that the variable x in $M[N, x.P]$ binds the free occurrences of x in P. The set of free variables of a λ_g-term M is denoted by $\mathsf{FV}(M)$. The symbol $\equiv$ denotes syntactical equality modulo α-conversion, and $\{\_/\_\}\_$ is used for usual capture-free substitution.

▶ **Definition 2** (Reduction system of λ_g)**.** The reduction rules are:

$$\begin{array}{lrcl} (\beta_g) & (\lambda x.M)[N, y.P] & \to & \{\{N/x\}M/y\}P \\ (\pi_g) & M[N, y.P][N', y'.P'] & \to & M[N, y.P[N', y'.P']] \end{array}$$

The reduction relation $\longrightarrow_{\beta_g,\pi_g}$ is defined by the contextual closure of the rules (β_g) and (π_g). We use $\longrightarrow^{+}_{\beta_g,\pi_g}$ for its transitive closure, and $\longrightarrow^{*}_{\beta_g,\pi_g}$ for its reflexive transitive closure. The set of λ_g-terms that are strongly normalising with respect to $\longrightarrow_{\beta_g,\pi_g}$ is denoted by $\mathsf{SN}^{\beta_g,\pi_g}$. These kinds of notations are also used for the notions of other reductions in this paper.

The type assignment system for λ_g-terms is defined by the rules in Figure 1. A *typing context* is defined as a finite set of pairs $\{x_1 : A_1, \ldots, x_n : A_n\}$ where the variables are pairwise distinct. The typing context $\Gamma, x : A$ denotes the union $\Gamma \cup \{x : A\}$ where x does not appear in Γ. We write $\Gamma \vdash_{\lambda_g} M : A$ if $\Gamma \vdash M : A$ is derivable with the rules of Figure 1. We also write $\Gamma \vdash_{\lambda} t : A$ if $\Gamma \vdash t : A$ is derivable with the standard rules of the simply typed λ-calculus.

$$\frac{}{\Gamma, x : A \vdash x : A}\ (\mathsf{Var}) \qquad \frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x.M : A \to B}\ (\mathsf{Abs})$$

$$\frac{\Gamma \vdash M : A \to B \quad \Gamma \vdash N : A \quad \Gamma, y : B \vdash P : C}{\Gamma \vdash M[N, y.P] : C}\ (\mathsf{GApp})$$

Figure 1 Type assignment system for λ_g-terms.

The reduction rules (β_g) and (π_g), when applied to typed terms, correspond to transformation of typing derivations. In Figure 2 we show the transformation corresponding to (π_g).

2.2 $\lambda I_{[,]}$-calculus

In this subsection we recall the definition and some properties of Klop's extended λ-calculus, which is referred to as $\lambda I_{[,]}$ in [18].

$$\dfrac{\dfrac{\Gamma \vdash M : A \to B \quad \Gamma \vdash N : A \quad \Gamma, y : B \vdash P : C \to D}{\Gamma \vdash M[N, y.P] : C \to D} \qquad \Gamma \vdash N' : C \quad \Gamma, y' : D \vdash P' : E}{\Gamma \vdash M[N, y.P][N', y'.P'] : E}$$

is transformed into

$$\dfrac{\Gamma \vdash M : A \to B \quad \Gamma \vdash N : A \quad \dfrac{\Gamma, y : B \vdash P : C \to D \quad \Gamma' \vdash N' : C \quad \Gamma', y' : D \vdash P' : E}{\Gamma, y : B \vdash P[N', y'.P'] : E}}{\Gamma \vdash M[N, y.P[N', y'.P']] : E}$$

where $\Gamma' = \Gamma, y : B$ and $y : B$ is added by weakening.

Figure 2 Derivation transformation corresponding to (π_g).

▶ **Definition 3** (Grammar of $\lambda I_{[,]}$)**.** The set $\Lambda I_{[,]}$ of terms of the $\lambda I_{[,]}$-calculus is defined by the following grammar:

$$T, U ::= x \mid \lambda x.T \mid T\,U \mid [T, U]$$

with the additional restriction that every abstraction $\lambda x.T$ satisfies $x \in \mathsf{FV}(T)$.

We denote lists of $\lambda I_{[,]}$-terms using vectors, and if $\overrightarrow{T} = T_1, \ldots, T_n$ then $[U, \overrightarrow{T}]$ denotes $[\ldots[U, T_1], \ldots, T_n]$ when $n \geq 1$, and U when $n = 0$.

The following property is straightforward by induction on terms.

▶ **Lemma 4** (Stability under substitution [18])**.**
If $T, U \in \Lambda I_{[,]}$, then $\{U/x\}T \in \Lambda I_{[,]}$.

▶ **Definition 5** (Reduction system of $\lambda I_{[,]}$)**.** The reduction rules are:

$$\begin{array}{lrcl} (\beta) & (\lambda x.T)\,U & \to & \{U/x\}T \\ (\pi) & [T, U]\,T' & \to & [T\,T', U] \end{array}$$

The following remark is straightforward [18]:

▶ **Lemma 6.** *If $T \longrightarrow_{\beta,\pi} T'$ then $\mathsf{FV}(T) = \mathsf{FV}(T')$ and $\{T/x\}U \longrightarrow^{+}_{\beta,\pi} \{T'/x\}U$ provided that $x \in \mathsf{FV}(U)$.*

Now we recall from [19, 20] an encoding of λ-calculus into $\lambda I_{[,]}$:

▶ **Definition 7** (Encoding of λ-calculus into $\lambda I_{[,]}$)**.** We encode the λ-calculus into $\lambda I_{[,]}$ as follows:

$$\begin{array}{lcll} \mathsf{i}(x) & := & x & \\ \mathsf{i}(\lambda x.t) & := & \lambda x.\mathsf{i}(t) & \text{if } x \in \mathsf{FV}(t) \\ \mathsf{i}(\lambda x.t) & := & \lambda x.[\mathsf{i}(t), x] & \text{if } x \notin \mathsf{FV}(t) \\ \mathsf{i}(t\,u) & := & \mathsf{i}(t)\mathsf{i}(u) & \end{array}$$

Note that this encoding is different from Klop's ι [18] in that the latter does not make the case distinction for abstractions.

A crucial property of the encoding, on which all strong normalisation results in this paper depend, is the following:

▶ **Theorem 8** ([19, 20])**.** *For any λ-term t, if $t \in \mathcal{SN}^{\beta}$ then $\mathsf{i}(t) \in \mathcal{SN}^{\beta,\pi}$.*

2.3 Strong normalisation of typed λ_g-terms

Our aim of this section is to show that all typed λ_g-terms are strong normalising with respect to β_g, π_g-reduction. The result has already been proved in several ways (e.g. [12, 22, 24]), but the strong normalisation results in later sections have not been obtained by those methods.

A naive attempt is to reduce the problem to the strong normalisation of β-reduction in the simply typed λ-calculus, using the translation $\mathcal{F}$ that maps all terms $M[N, y.P]$ into $\{\mathcal{F}(M)\mathcal{F}(N)/y\}\mathcal{F}(P)$. However, this translation does not necessarily preserve one or more reduction steps; for instance, if $M \longrightarrow_{\beta_g,\pi_g} M'$ then $M[N, y.z] \longrightarrow_{\beta_g,\pi_g} M'[N, y.z]$, but $\mathcal{F}(M[N, y.z]) \equiv z \equiv \mathcal{F}(M'[N, y.z])$. So for our purpose some modification of the translation is needed. Here we introduce the following one, taking account of the free occurrences of y in P for $M[N, y.P]$.

▶ **Definition 9** (Encoding of λ_g into λ-calculus)**.** We encode the λ_g into λ-calculus as follows:

$$\begin{array}{lcll} \mathcal{G}(x) & := & x & \\ \mathcal{G}(\lambda x.M) & := & \lambda x.\mathcal{G}(M) & \\ \mathcal{G}(M[N, y.P]) & := & \{\mathcal{G}(M)\mathcal{G}(N)/y\}\mathcal{G}(P) & \text{if } y \in \mathsf{FV}(P) \\ \mathcal{G}(M[N, y.P]) & := & (\lambda y.\mathcal{G}(P))(\mathcal{G}(M)\mathcal{G}(N)) & \text{if } y \notin \mathsf{FV}(P) \end{array}$$

Unfortunately, this encoding does not allow simulation of reduction.

▶ **Example 10.** Let $M_1 \equiv m[n, y.(\lambda x.z)[y[z, w.w], v.v]]$ and $N_1 \equiv m[n, y.z]$. Then $M_1 \longrightarrow_{\beta_g} N_1$ holds, but for their encodings $\mathcal{G}(M_1) \equiv (\lambda x.z)(mnz)$ and $\mathcal{G}(N_1) \equiv (\lambda y.z)(mn)$, the former cannot reduce to the latter.

The above encoding will be used not for simulation of reduction but for the lifting of a λ_g-term to be proved strongly normalising. For simulation, we use as the target calculus $\lambda I_{[,]}$ instead of λ-calculus, and the translation is now defined to be non-deterministic. In Figure 3 we give the inductive definition of the relation $\mathcal{H}$ between λ_g-terms and $\lambda I_{[,]}$-terms.

$$\frac{}{x \;\mathcal{H}\; x} \qquad \frac{M \;\mathcal{H}\; T \quad N \;\mathcal{H}\; U \quad P \;\mathcal{H}\; S}{M[N, y.P] \;\mathcal{H}\; \{T\,U/y\}S}\; y \in \mathsf{FV}(S)$$

$$\frac{M \;\mathcal{H}\; T}{\lambda x.M \;\mathcal{H}\; \lambda x.T}\; x \in \mathsf{FV}(T) \qquad \frac{M \;\mathcal{H}\; T}{M \;\mathcal{H}\; [T, U]}\; U \in \Lambda I_{[,]}$$

Figure 3 Relation between λ_g & $\lambda I_{[,]}$.

▶ **Lemma 11.** *If $M \;\mathcal{H}\; T$, then*

1. $\mathsf{FV}(M) \subseteq \mathsf{FV}(T)$
2. $T \in \Lambda I_{[,]}$
3. $x \notin \mathsf{FV}(M)$ *and* $U \in \Lambda I_{[,]}$ *implies* $M \;\mathcal{H}\; \{U/x\}T$
4. $\{y/x\}M \;\mathcal{H}\; \{y/x\}T$

▶ **Example 12.** $M_1 \equiv m[n, y.(\lambda x.z)[y[z, w.w], v.v]] \;\mathcal{H}\; (\lambda x.[z, x])(mnz)$ and $N_1 \equiv m[n, y.z] \;\mathcal{H}\; [z, mnz]$ as shown in Figures 4 and 5. Note that $(\lambda x.[z, x])(mnz)$ β-reduces to $[z, mnz]$ in contrast with the encodings in Example 10. The point is that yz, which corresponds to $y[z, w.w]$ discarded by β_g-reduction from M_1, is retained in the $\lambda I_{[,]}$-term $[z, yz]$ in Figure 5.

$$\dfrac{\dfrac{}{m \mathrel{\mathcal{H}} m} \quad \dfrac{}{n \mathrel{\mathcal{H}} n} \quad \dfrac{\dfrac{\dfrac{}{z \mathrel{\mathcal{H}} z}}{z \mathrel{\mathcal{H}} [z,x]}}{\lambda x.z \mathrel{\mathcal{H}} \lambda x.[z,x]} \quad \dfrac{\dfrac{}{y \mathrel{\mathcal{H}} y} \quad \dfrac{}{z \mathrel{\mathcal{H}} z} \quad \dfrac{}{w \mathrel{\mathcal{H}} w}}{y[z,w.w] \mathrel{\mathcal{H}} \{yz/w\}w} \quad \dfrac{}{v \mathrel{\mathcal{H}} v}}{(\lambda x.z)[y[z,w.w],v.v] \mathrel{\mathcal{H}} \{(\lambda x.[z,x])(yz)/v\}v}}{m[n,y.(\lambda x.z)[y[z,w.w],v.v]] \mathrel{\mathcal{H}} \{mn/y\}((\lambda x.[z,x])(yz))}$$

Figure 4 Derivation of $m[n,y.(\lambda x.z)[y[z,w.w],v.v]] \mathrel{\mathcal{H}} (\lambda x.[z,x])(mnz)$.

$$\dfrac{\dfrac{}{m \mathrel{\mathcal{H}} m} \quad \dfrac{}{n \mathrel{\mathcal{H}} n} \quad \dfrac{\dfrac{}{z \mathrel{\mathcal{H}} z}}{z \mathrel{\mathcal{H}} [z,yz]}}{m[n,y.z] \mathrel{\mathcal{H}} \{mn/y\}[z,yz]}$$

Figure 5 Derivation of $m[n,y.z] \mathrel{\mathcal{H}} [z,mnz]$.

Now our aim is to show that reduction in λ_g is simulated in $\lambda I_{[,]}$ through $\mathcal{H}$. For this we need the following lemma.

▶ **Lemma 13.** *If $M \mathrel{\mathcal{H}} T$ and $N \mathrel{\mathcal{H}} U$, then $\{N/x\}M \mathrel{\mathcal{H}} \{U/x\}T$.*

Proof. By induction on the derivation of $M \mathrel{\mathcal{H}} T$. Here we only consider the case where the last applied rule of the derivation is

$$\dfrac{M' \mathrel{\mathcal{H}} T' \quad N' \mathrel{\mathcal{H}} U' \quad P \mathrel{\mathcal{H}} S}{M'[N',y.P] \mathrel{\mathcal{H}} \{T'U'/y\}S} \; y \in \mathsf{FV}(S)$$

Then we have

$$\dfrac{\overset{\text{I.H.}}{\{N/x\}M' \mathrel{\mathcal{H}} \{U/x\}T'} \quad \overset{\text{I.H.}}{\{N/x\}N' \mathrel{\mathcal{H}} \{U/x\}U'} \quad \overset{\text{I.H.}}{\{N/x\}P \mathrel{\mathcal{H}} \{U/x\}S}}{\begin{array}{c}\{N/x\}M'[\{N/x\}N',y.\{N/x\}P] \mathrel{\mathcal{H}} \{\{U/x\}T'\{U/x\}U'/y\}\{U/x\}S \\ \| \\ \{N/x\}(M'[N',y.P]) \mathrel{\mathcal{H}} \{U/x\}\{T'U'/y\}S\end{array}}$$

◀

Now we are in a position to prove the simulation theorem in $\lambda I_{[,]}$.

▶ **Theorem 14** (Simulation in $\lambda I_{[,]}$)**.** *Suppose $M \mathrel{\mathcal{H}} T$.*

1. *If $M \longrightarrow_{\beta_g} N$ then there exists U such that $N \mathrel{\mathcal{H}} U$ and $T \longrightarrow^{+}_{\beta,\pi} U$.*
2. *If $M \longrightarrow_{\pi_g} N$ then $N \mathrel{\mathcal{H}} T$.*

Proof. By induction on the derivation of $M \mathrel{\mathcal{H}} T$.

- The case of the rule
$$\dfrac{}{x \mathrel{\mathcal{H}} x}$$
is vacuous.
- For the rule
$$\dfrac{M \mathrel{\mathcal{H}} T}{M \mathrel{\mathcal{H}} [T,U]} \; U \in \Lambda I_{[,]}$$
we simply apply the induction hypothesis.

- For the rule
$$\frac{M \mathrel{\mathcal{H}} T}{\lambda x.M \mathrel{\mathcal{H}} \lambda x.T}\ x \in \mathsf{FV}(T)$$
the reduction must take place within M, so we can apply the induction hypothesis, remembering that reduction in $\lambda I_{[,]}$ preserves free variables (Lemma 6), so the side-condition remains satisfied.
- The interesting case is
$$\frac{M \mathrel{\mathcal{H}} T \quad N \mathrel{\mathcal{H}} U \quad P \mathrel{\mathcal{H}} S}{M[N, y.P] \mathrel{\mathcal{H}} \{T\,U/y\}S}\ y \in \mathsf{FV}(S)$$
If the reduction takes place within M, N or P, we apply the induction hypothesis again, and the side-condition remains satisfied. Moreover, a β_g-reduction step in M or N is simulated by at least one reduction step from T or U and that step is preserved in the reduction of $\{T\,U/y\}S$ since $y \in \mathsf{FV}(S)$.
Otherwise, the reduction takes place at the root. We inspect the two cases, noting that the form of the last part of the derivation is determined by the redex.
1. $(\lambda x.M)[N, y.P] \longrightarrow_{\beta_g} \{\{N/x\}M/y\}P$. Then the derivation has the form
$$\dfrac{\dfrac{\dfrac{M \mathrel{\mathcal{H}} T}{\lambda x.M \mathrel{\mathcal{H}} \lambda x.T}\ x \in \mathsf{FV}(T)}{\lambda x.M \mathrel{\mathcal{H}} [\lambda x.T, \overrightarrow{R}]} \quad N \mathrel{\mathcal{H}} U \quad P \mathrel{\mathcal{H}} S}{(\lambda x.M)[N, y.P] \mathrel{\mathcal{H}} \{[\lambda x.T, \overrightarrow{R}]U/y\}S}\ y \in \mathsf{FV}(S)$$
By applying Lemma 13 twice, we have
$$\dfrac{\dfrac{\dfrac{N \mathrel{\mathcal{H}} U \quad M \mathrel{\mathcal{H}} T}{\{N/x\}M \mathrel{\mathcal{H}} \{U/x\}T}\ \text{Lemma 13}}{\{N/x\}M \mathrel{\mathcal{H}} [\{U/x\}T, \overrightarrow{R}]} \quad P \mathrel{\mathcal{H}} S}{\{\{N/x\}M/y\}P \mathrel{\mathcal{H}} \{[\{U/x\}T, \overrightarrow{R}]/y\}S}\ \text{Lemma 13}$$
Since $y \in \mathsf{FV}(S)$, we have $\{[\lambda x.T, \overrightarrow{R}]U/y\}S \longrightarrow^{+}_{\beta,\pi} \{[\{U/x\}T, \overrightarrow{R}]/y\}S$ as required.
2. $M[N, y.P][N', y'.P'] \longrightarrow_{\pi_g} M[N, y.P[N', y'.P']]$. In this case, the derivation has the form
$$\dfrac{\dfrac{\dfrac{M \mathrel{\mathcal{H}} T \quad N \mathrel{\mathcal{H}} U \quad P \mathrel{\mathcal{H}} S}{M[N, y.P] \mathrel{\mathcal{H}} \{T\,U/y\}S}\ y \in \mathsf{FV}(S)}{M[N, y.P] \mathrel{\mathcal{H}} [\{T\,U/y\}S, \overrightarrow{R}]} \quad N' \mathrel{\mathcal{H}} U' \quad P' \mathrel{\mathcal{H}} S'}{M[N, y.P][N', y'.P'] \mathrel{\mathcal{H}} \{[\{T\,U/y\}S, \overrightarrow{R}]U'/y'\}S'}\ y' \in \mathsf{FV}(S')$$
Then we have
$$\dfrac{M \mathrel{\mathcal{H}} T \quad N \mathrel{\mathcal{H}} U \quad \dfrac{\dfrac{P \mathrel{\mathcal{H}} S}{P \mathrel{\mathcal{H}} [S, \overrightarrow{R}]} \quad N' \mathrel{\mathcal{H}} U' \quad P' \mathrel{\mathcal{H}} S'}{P[N', y'.P'] \mathrel{\mathcal{H}} \{[S, \overrightarrow{R}]U'/y'\}S'}\ y' \in \mathsf{FV}(S')}{M[N, y.P[N', y'.P']] \mathrel{\mathcal{H}} \{T\,U/y\}\{[S, \overrightarrow{R}]U'/y'\}S'}\ y \in \mathsf{FV}(S'')$$
$$\Vert\!\Vert$$
$$M[N, y.P[N', y'.P']] \mathrel{\mathcal{H}} \{[\{T\,U/y\}S, \overrightarrow{R}]U'/y'\}S'$$
where $S'' \equiv \{[S, \overrightarrow{R}]U'/y'\}S'$. Since $y' \in \mathsf{FV}(S')$ and $y \in \mathsf{FV}(S)$, we have $y \in \mathsf{FV}(S'')$.
◀

To prove the strong normalisation of any typed λ_g-term, we lift it to a $\lambda I_{[,]}$-term through the encodings $\mathcal{G}$ and i.

▶ **Lemma 15.** *For any λ_g-term M, there exists a $\lambda I_{[,]}$-term T such that $M \mathrel{\mathcal{H}} T$ and $\mathsf{i}(\mathcal{G}(M)) \longrightarrow^*_{\beta,\pi} T$.*

Proof. By induction on M. (For the details, see Appendix A.) ◀

Finally we show that π_g-reduction is strongly normalising.

▶ **Lemma 16.** *$\longrightarrow_{\pi_g}$ is strongly normalising.*

Proof. We define a map $h : \Lambda_g \longrightarrow \mathbb{N}$ as follows: $h(x) := 1$, $h(\lambda x.M) := h(M)$, and $h(M[N, y.P]) := h(M) \times (h(N) + h(P))$. Then observe that if $M \longrightarrow_{\pi_g} N$ then $h(M) > h(N)$. ◀

Now we can prove the strong normalisation theorem of typed λ_g-terms.

▶ **Theorem 17** (Strong normalisation)**.**
For any λ_g-term M, if $\Gamma \vdash_{\lambda_g} M : A$ then $M \in \mathit{SN}^{\beta_g,\pi_g}$.

Proof. Suppose there is an infinite β_g, π_g-reduction sequence from M. Since π_g-reduction is strongly normalising (Lemma 16), the sequence has infinitely many β_g-reduction steps.

Now, from $\Gamma \vdash_{\lambda_g} M : A$, we have $\Gamma \vdash_{\lambda} \mathcal{G}(M) : A$, so by the strong normalisation of typed λ-terms, $\mathcal{G}(M) \in \mathsf{SN}^{\beta}$. Hence by Theorem 8, $\mathsf{i}(\mathcal{G}(M)) \in \mathsf{SN}^{\beta,\pi}$.

By Lemma 15, there is a $\lambda I_{[,]}$-term T such that $M \mathrel{\mathcal{H}} T$ and $\mathsf{i}(\mathcal{G}(M)) \longrightarrow^*_{\beta,\pi} T$. Then, applying Theorem 14 to each β_g, π_g-reduction step of the infinite reduction sequence from M, we have an infinite β, π-reduction sequence

$$T \longrightarrow^+_{\beta,\pi} T_1 \longrightarrow^+_{\beta,\pi} T_2 \longrightarrow^+_{\beta,\pi} \cdots$$

which is a contradiction. ◀

3 Strong normalisation for $\lambda \mathsf{x}_g$-calculus

In the following we extend the syntax of λ_g-calculus by explicit substitution and study properties of the calculus. Strong normalisation of typed terms is proved using an extension of the non-deterministic translation in the previous section.

3.1 $\lambda \mathsf{x}_g$-calculus

In this subsection we introduce a modification of the explicit substitution calculus in [22], which we call $\lambda \mathsf{x}_g$-calculus. As shown in [22], typed terms of the calculus are isomorphic to proofs in intuitionistic sequent calculus modulo a term quotient. First we define the syntax of the type-free calculus.

▶ **Definition 18** (Grammar of $\lambda \mathsf{x}_g$)**.** The set $\Lambda \mathsf{x}_g$ of terms of the $\lambda \mathsf{x}_g$-calculus is defined by the following grammar:

$$M, N, P ::= x \mid \lambda x.M \mid M[N, x.P] \mid \langle M/x \rangle N$$

The notions of free and bound variables are extended from those for λ_g by the clause that the variable x in $\langle M/x \rangle N$ binds the free occurrences of x in N.

▶ **Definition 19** (Reduction system of $\lambda \mathsf{x}_g$)**.** The reduction rules are:

(1)	$\langle M/x\rangle y \to y$	$(x \not\equiv y)$
(2)	$\langle M/x\rangle x \to M$	
(3)	$\langle M/x\rangle(\lambda y.N) \to \lambda y.\langle M/x\rangle N$	
(4)	$\langle M/x\rangle(y[N,z.P]) \to y[\langle M/x\rangle N, z.\langle M/x\rangle P]$	$(x \not\equiv y)$
(5)	$\langle M/x\rangle(x[N,z.P]) \to M[\langle M/x\rangle N, z.\langle M/x\rangle P]$	$(x \in \mathsf{FV}([N,z.P]))$
(6)	$\langle M/x\rangle(Q[N,z.P]) \to (\langle M/x\rangle Q)[\langle M/x\rangle N, z.\langle M/x\rangle P]$	(Q is not a variable)
(7)	$\langle M/x\rangle(x[N,z.P]) \to M[N,z.P]$	$(x \notin \mathsf{FV}([N,z.P]))$
(B_1)	$(\lambda y.M)[N,z.P] \to \langle N/y\rangle\langle M/z\rangle P$	
(B_2)	$(\lambda y.M)[N,z.P] \to \langle\langle N/y\rangle M/z\rangle P$	
(Pi)	$M[N,z.P][N',z'.P'] \to M[N,z.P[N',z'.P']]$	

The reduction relation $\longrightarrow_{\lambda \mathsf{x}_g}$ is defined by the contextual closure of all the reduction rules. We define two subsystems of $\lambda \mathsf{x}_g$: the system B consists of the rules (B_1) and (B_2), and the system x consists of the rules (1)-(7) and (Pi).

▶ Remark. The reduction rules of Λ_{gx} in [22] are the rules (1)-(7), (B_1) and the following:

$$(\mathsf{Pi}') \quad M[N,z.P][N',z'.P'] \to M[N,z.\langle P/x\rangle(x[N',z'.P'])] \quad (x \notin \mathsf{FV}([N',z'.P']))$$

Note that the system x in [22] does not include the above rule (Pi′), while our system x includes the rule (Pi).

The type assignment system for $\lambda \mathsf{x}_g$-terms is defined by the rules in Figure 1 and the following:

$$\frac{\Gamma \vdash M : A \qquad \Gamma, x : A \vdash N : B}{\Gamma \vdash \langle M/x\rangle N : B}\ (\mathsf{Sub})$$

We write $\Gamma \vdash_{\lambda \mathsf{x}_g} M : A$ if $\Gamma \vdash M : A$ is derivable with those rules.

When applied to typed terms, the reduction rules correspond to transformation of typing derivations. In Figure 6 we show the transformation corresponding to (B_1) and (B_2).

$$\frac{\dfrac{\Gamma, y : A \vdash M : B}{\Gamma \vdash \lambda y.M : A \to B} \qquad \Gamma \vdash N : A \qquad \Gamma, z : B \vdash P : C}{\Gamma \vdash (\lambda y.M)[N,z.P] : C}$$

is transformed into

$$\frac{\Gamma \vdash N : A \qquad \dfrac{\Gamma, y : A \vdash M : B \qquad \Gamma, y : A, z : B \vdash P : C}{\Gamma, y : A \vdash \langle M/z\rangle P : B}}{\Gamma \vdash \langle N/y\rangle\langle M/z\rangle P : C}\ \text{by } (\mathsf{B}_1)$$

and

$$\frac{\dfrac{\Gamma \vdash N : A \qquad \Gamma, y : A \vdash M : B}{\Gamma \vdash \langle N/y\rangle M : B} \qquad \Gamma, z : B \vdash P : C}{\Gamma \vdash \langle\langle N/y\rangle M/z\rangle P : C}\ \text{by } (\mathsf{B}_2)$$

Figure 6 Derivation transformation corresponding to (B_1) and (B_2).

3.2 Failure of PSN with respect to β_g, π_g-reduction

The main result of this paper is the strong normalisation theorem of typed $\lambda \mathsf{x}_g$-terms. It has been proved by Nakazawa [22] for the case where the reduction system does not include the

rule (B_2). He also mentioned the difficulty in proving strong normalisation in the presence of (B_2). In this subsection we explain why the method in [22] does not work for the system with (B_2).

A standard method of proving strong normalisation of explicit substitution calculi [6, 4, 5] uses projection onto normal forms of the substitution subcalculus. Those normal forms are terms without explicit substitution, and the proof relies on the strong normalisation result of the original calculus without explicit substitution. Such a proof works even for type-free calculi to show the property called Preservation of Strong Normalisation (PSN), which states that if a term is strongly normalising with respect to reduction of the original calculus then it is also strongly normalising in the explicit substitution calculus. However, this property does not hold between λ_g-calculus and $\lambda\mathsf{x}_g$-calculus.

▶ **Example 20.** $\lambda\mathsf{x}_g$-calculus does not satisfy PSN with respect to β_g, π_g-reduction as the following example shows. Let $\omega \equiv \lambda y.y[y, v.v]$. Then

$$\omega[\omega, z.x] \longrightarrow_{\beta_g} \{\{\omega/y\}(y[y, v.v])/z\}x \equiv x$$

Since this is the only β_g, π_g-reduction sequence from $\omega[\omega, z.x]$, it is in $\mathsf{SN}^{\beta_g,\pi_g}$. However,

$$\begin{array}{ll} \omega[\omega, z.x] & \longrightarrow_{\mathsf{B}_2} \langle\langle\omega/y\rangle(y[y, v.v])/z\rangle x \\ & \longrightarrow^*_{\lambda\mathsf{x}_g} \langle\omega[\omega, v.v]/z\rangle x \\ & \longrightarrow_{\mathsf{B}_2} \cdots \end{array}$$

Hence $\omega[\omega, z.x] \notin \mathsf{SN}^{\lambda\mathsf{x}_g}$.

In spite of the above fact, strong normalisation of typed $\lambda\mathsf{x}_g$-terms may be proved, but then one cannot use a proof method that would yield at the same time PSN of the type-free calculus with respect to β_g, π_g-reduction. Specifically, a standard method as in [22], which projects $\lambda\mathsf{x}_g$-terms onto λ_g-terms and relies on the result of strong normalisation of β_g, π_g-reduction, does not work.

▶ Remark. An intended meaning of the β_g-rule $(\lambda y.M)[N, z.P] \to \{\{N/y\}M/z\}P$ of λ_g-calculus is that the function $\lambda y.M$ is applied to the argument N and then the result of the application is passed to the continuation $z.P$. In the type-free case, the computation of the application may not produce any result as seen in the example above, but even so, the term $\{N/y\}M$ is substituted for z in P; in particular, when z does not occur free in P, the term $\{N/y\}M$ is discarded. This means that λ_g-calculus can not express a natural operational semantics that passes to the continuation the result of the application after computing it, but only implement some specific strategies. On the other hand, $\lambda\mathsf{x}_g$-calculus and other formalisms like $\overline{\lambda}\mu\widetilde{\mu}$ [7] allow for such a natural operational semantics. (Those calculi do not satisfy PSN with respect to β_g, π_g-reduction, but they satisfy PSN with respect to β-reduction in an isomorphic image of the λ-calculus through appropriate embeddings.)

3.3 Strong normalisation of typed $\lambda\mathsf{x}_g$-terms

Our proof of strong normalisation of typed $\lambda\mathsf{x}_g$-terms proceeds in a similar pattern to Section 2 except for the treatment of explicit substitution. To deal with explicit substitution we use another technique from [19, 20] with the notions of safe and minimal reductions.

▶ **Definition 21.** A reduction step is *minimal* if every proper subterm of the redex is in $\mathsf{SN}^{\lambda\mathsf{x}_g}$. A minimal reduction step is *safe* if the redex itself is in $\mathsf{SN}^{\lambda\mathsf{x}_g}$, and *unsafe* if not.

We can restrict infinite $\lambda \mathsf{x}_g$-reduction sequences to those consisting only of minimal reduction steps.

▶ **Lemma 22.** *If $M \notin \mathit{SN}^{\lambda \mathsf{x}_g}$ then there exists an infinite $\lambda \mathsf{x}_g$-reduction sequence starting from M such that all the reduction steps are minimal.*

Proof. It suffices to take in each reduction step an innermost redex of the ones that preserve the possibility of infinite reduction. (Such a reduction sequence is called a minimal infinite reduction sequence, e.g. in [3].) ◀

▶ **Definition 23.** Let h be a subsystem of $\lambda \mathsf{x}_g$, and let $M \longrightarrow_{\mathsf{h}} N$. We write $M \longrightarrow_{\mathsf{minh}} N$ (resp. $M \longrightarrow_{\mathsf{safeh}} N$) to denote that the reduction step $M \longrightarrow_{\mathsf{h}} N$ is minimal (resp. safe) (where minimality is with respect to $\mathsf{SN}^{\lambda \mathsf{x}_g}$ and not for the subsystem h).

A crucial point of our proof is that we divide minimal reduction steps into two kinds: One is those which are simulated in $\lambda I_{[,]}$ so that one or more reduction steps are preserved, as β_g-reduction steps in Section 2. The other is those which are strongly normalising in $\lambda \mathsf{x}_g$ and simulated in $\lambda I_{[,]}$ where one or more reduction steps are not necessarily preserved, as π_g-reduction steps in Section 2. In this section we take unsafe B-reduction steps as the former and the rest (i.e. reduction steps by $\longrightarrow_{\mathsf{safeB,minx}}$) as the latter.

To show that $\longrightarrow_{\mathsf{safeB,minx}}$ is strongly normalising, we briefly recall the lexicographic path ordering [13]. For a more detailed description and proofs, the reader is referred to, e.g. [2].

▶ **Definition 24** (Lexicographic path ordering)**.** Let $\succ$ be a transitive and irreflexive ordering on the set of function symbols in a first-order signature, and let $s \equiv f(s_1, \ldots, s_m)$ and $t \equiv g(t_1, \ldots, t_n)$ be terms over the signature. Then $s >_{\mathsf{lpo}} t$, if one of the following holds:

1. $s_i \equiv t$ or $s_i >_{\mathsf{lpo}} t$ for some $i = 1, \ldots, m$,
2. $f \succ g$ and $s >_{\mathsf{lpo}} t_j$ for all $j = 1, \ldots, n$,
3. $f \equiv g$, $s >_{\mathsf{lpo}} t_j$ for all $j = 1, \ldots, n$, and $s_1 \equiv t_1, \ldots, s_{i-1} \equiv t_{i-1}, s_i >_{\mathsf{lpo}} t_i$ for some $i = 1, \ldots, m$.

▶ **Theorem 25.** *$>_{\mathsf{lpo}}$ is well-founded if and only if $\succ$ is well-founded.*

Now we encode $\lambda \mathsf{x}_g$-terms into a first-order syntax given by the following ordered infinite signature:

$$\mathsf{sub}(\_,\_) \succ \mathsf{gapp}(\_,\_,\_) \succ \mathsf{abs}(\_) \succ \mathsf{c}^{(m,n)}$$

where for every $m, n \in \mathbb{N}$, there is a constant $\mathsf{c}^{(m,n)}$. Those constants are all below $\mathsf{abs}(\_)$, and the precedence between them is given by $\mathsf{c}^{(m,n)} \succ \mathsf{c}^{(m',n')}$ if $(m,n) > (m',n')$ lexicographically. Then the precedence relation is well-founded, and so $>_{\mathsf{lpo}}$ induced on the first-order terms is also well-founded.

Let M be a $\lambda \mathsf{x}_g$-term with $M \in \mathsf{SN}^{\lambda \mathsf{x}_g}$. We define $w(M)$ as $(\mathsf{maxred}(M), |M|)$ where $\mathsf{maxred}(M)$ is the maximal length of all $\lambda \mathsf{x}_g$-reduction sequences starting from M, and $|M|$ is the size of M. Then the aforementioned encoding is given in Figure 7.

▶ **Lemma 26.** *If $M \longrightarrow_{\mathsf{safeB,minx}} M'$ then $\overline{M} >_{\mathsf{lpo}} \overline{M'}$. Hence, $\longrightarrow_{\mathsf{safeB,minx}}$ is strongly normalising.*

Proof. By induction on the derivation of the reduction step. For the details, see Appendix B. ◀

The relation $\mathcal{H}$ between $\lambda \mathsf{x}_g$-terms and $\lambda I_{[,]}$-terms is inductively defined by the rules in Figure 3 and the following:

$$\frac{M \mathrel{\mathcal{H}} T \qquad N \mathrel{\mathcal{H}} U}{\langle M/x \rangle N \mathrel{\mathcal{H}} \{T/x\}U} \; x \in \mathsf{FV}(U) \vee M \in \mathsf{SN}^{\lambda \mathsf{x}_g}$$

$$
\begin{array}{lcll}
\overline{M} & := & \mathsf{c}^{w(M)} & \text{if } M \in \mathsf{SN}^{\lambda \mathsf{x}_g} \\
\text{otherwise} & & & \\
\overline{\lambda x.M} & := & \mathsf{abs}(\overline{M}) & \\
\overline{M[N, y.P]} & := & \mathsf{gapp}(\overline{M}, \overline{N}, \overline{P}) & \\
\overline{\langle M/x \rangle N} & := & \mathsf{sub}(\overline{M}, \overline{N}) &
\end{array}
$$

Figure 7 Encoding of $\lambda \mathsf{x}_g$ into a first-order syntax.

The side-condition of the above rule is designed so that an unsafe B-reduction step in M is simulated by at least one reduction step in $\{T/x\}U$ (cf. the first paragraph of page 413). It is also closely related to the notion of decent term in [27, Definition 4.4]. Note that in the presence of the above rule, Lemma 11 (0a) no longer holds.

▶ **Theorem 27** (Simulation in $\lambda I_{[,]}$)**.** *Suppose $M \mathrel{\mathcal{H}} T$.*

1. *If $M \longrightarrow_{\mathit{minB}} N$ and the reduction step is unsafe then there exists U such that $N \mathrel{\mathcal{H}} U$ and $T \longrightarrow^{+}_{\beta,\pi} U$.*
2. *If $M \longrightarrow_{\mathit{minB}} N$ and the reduction step is safe then there exists U such that $N \mathrel{\mathcal{H}} U$ and $T \longrightarrow^{*}_{\beta,\pi} U$.*
3. *If $M \longrightarrow_{\mathit{minx}} N$ then $N \mathrel{\mathcal{H}} T$.*

Proof. By induction on the derivation of $M \mathrel{\mathcal{H}} T$. A detailed proof is found in Appendix B. ◀

▶ **Definition 28** (Encoding of $\lambda \mathsf{x}_g$ into λ-calculus)**.** We encode $\lambda \mathsf{x}_g$ into λ-calculus, extending the definition of $\mathcal{G}$ (Definition 9) by

$$
\begin{array}{lcll}
\mathcal{G}(\langle M/x \rangle N) & := & \{\mathcal{G}(M)/x\}\mathcal{G}(N) & \text{if } x \in \mathsf{FV}(N) \\
\mathcal{G}(\langle M/x \rangle N) & := & (\lambda x.\mathcal{G}(N))\,\mathcal{G}(M) & \text{if } x \notin \mathsf{FV}(N)
\end{array}
$$

As in the previous section, we lift any $\lambda \mathsf{x}_g$-term to a $\lambda I_{[,]}$-term through $\mathcal{G}$ and i.

▶ **Lemma 29.** *For any $\lambda \mathsf{x}_g$-term M, there exists a $\lambda I_{[,]}$-term T such that $M \mathrel{\mathcal{H}} T$ and $\mathsf{i}(\mathcal{G}(M)) \longrightarrow^{*}_{\beta,\pi} T$.*

Proof. By induction on M. (For the details, see Appendix B.) ◀

Now we can prove the strong normalisation theorem of typed $\lambda \mathsf{x}_g$-terms.

▶ **Theorem 30** (Strong normalisation)**.**
For any $\lambda \mathsf{x}_g$-term M, if $\Gamma \vdash_{\lambda \mathsf{x}_g} M : A$ then $M \in \mathit{SN}^{\lambda \mathsf{x}_g}$.

Proof. Suppose $M \notin \mathsf{SN}^{\lambda \mathsf{x}_g}$. Then by Lemma 22, there exists an infinite $\lambda \mathsf{x}_g$-reduction sequence starting from M such that all the reduction steps are minimal. Since $\longrightarrow_{\mathsf{safeB},\mathsf{minx}}$ is strongly normalising (Lemma 26), the sequence has infinitely many unsafe B-reduction steps.

Now, from $\Gamma \vdash_{\lambda \mathsf{x}_g} M : A$, we have $\Gamma \vdash_{\lambda} \mathcal{G}(M) : A$, so by the strong normalisation of typed λ-terms, $\mathcal{G}(M) \in \mathsf{SN}^{\beta}$. Hence by Theorem 8, $\mathsf{i}(\mathcal{G}(M)) \in \mathsf{SN}^{\beta,\pi}$.

By Lemma 29, there is a $\lambda I_{[,]}$-term T such that $M \mathrel{\mathcal{H}} T$ and $\mathsf{i}(\mathcal{G}(M)) \longrightarrow^{*}_{\beta,\pi} T$. Then, applying Theorem 27 to each minimal reduction step of the infinite $\lambda \mathsf{x}_g$-reduction sequence from M, we have an infinite β,π-reduction sequence, which is a contradiction. ◀

4 Application to other systems

The proof method in the previous section provides a general framework for showing strong normalisation of systems on $\lambda\mathsf{x}_g$-terms with various reduction rules. In this section we illustrate that with an extension of Sørensen and Urzyczyn's cut-elimination system in intuitionistic sequent calculus [27].

▶ **Definition 31** (Reduction system of $\lambda\mathsf{x}_g^{\mathsf{SU}}$)**.** The reduction rules of $\lambda\mathsf{x}_g^{\mathsf{SU}}$ are the rules (1)-(4) of $\lambda\mathsf{x}_g$ (Definition 19) and the following:

(8) $\langle y/x\rangle(x[N,z.P]) \to \langle y/x\rangle(y[N,z.P])$
(9) $\langle y[N,z.P]/x\rangle(x[N',z'.P']) \to y[N,z.\langle P/x\rangle(x[N',z'.P'])]$
(B_3) $\langle \lambda y.M/x\rangle(x[N,z.P]) \to \langle \lambda y.M/x\rangle\langle\langle N/y\rangle M/z\rangle P$

The reduction relation $\longrightarrow_{\lambda\mathsf{x}_g^{\mathsf{SU}}}$ is defined by the contextual closure of those reduction rules. The subsystem x^{SU} consists of the rules (1)-(4), (8) and (9).

▶ Remark. The reduction rules of the system in [27, page 920] are the same as those of $\lambda\mathsf{x}_g^{\mathsf{SU}}$, but the terms are restricted to those such that M is a variable in $M[N,y.P]$.

The notions of minimal, safe and unsafe reductions and the encoding into the first-order syntax are defined as in the case of $\lambda\mathsf{x}_g$. We define $d(M)$ as the number of subterms of M that have the form $\langle y/x\rangle(x[N,z.P])$. Then we can prove the following lemma and the simulation theorem.

▶ **Lemma 32.** *Let* $\mathsf{h} := \mathsf{safeB}_3, \mathsf{minx}^{\mathsf{SU}}$. *If* $M \longrightarrow_{\mathsf{h}} M'$ *then* $\overline{M} >_{\mathsf{lpo}} \overline{M'}$ *or* $\overline{M} = \overline{M'}$ *and* $d(M) > d(M')$. *Hence,* $\longrightarrow_{\mathsf{h}}$ *is strongly normalising.*

Proof. By induction on the derivation of the reduction step. ◀

▶ **Theorem 33** (Simulation in $\lambda I_{[,]}$)**.** *Suppose* $M \mathcal{H} T$.

1. *If* $M \longrightarrow_{\mathsf{minB}_3} N$ *and the reduction step is unsafe then there exists* U *such that* $N \mathcal{H} U$ *and* $T \longrightarrow^{+}_{\beta,\pi} U$.
2. *If* $M \longrightarrow_{\mathsf{minB}_3} N$ *and the reduction step is safe then there exists* U *such that* $N \mathcal{H} U$ *and* $T \longrightarrow^{*}_{\beta,\pi} U$.
3. *If* $M \longrightarrow_{\mathsf{minx}^{\mathsf{SU}}} N$ *then* $N \mathcal{H} T$.

Proof. By induction on the derivation of $M \mathcal{H} T$. ◀

Using the above lemma and theorem, we can prove strong normalisation of typed $\lambda\mathsf{x}_g$-terms with respect to reduction of $\lambda\mathsf{x}_g^{\mathsf{SU}}$.

▶ **Theorem 34** (Strong normalisation)**.**
For any $\lambda\mathsf{x}_g$*-term* M, *if* $\Gamma \vdash_{\lambda\mathsf{x}_g} M : A$ *then* $M \in \mathcal{SN}^{\lambda\mathsf{x}_g^{\mathsf{SU}}}$.

Proof. Similar to the proof of Theorem 30. ◀

Since proof terms for intuitionistic sequent calculus have the same type in the type assignment system of [27] and in ours, it follows that the cut-elimination procedure is strongly normalising.

As we have seen, in our framework, proving strong normalisation of systems with various reduction rules on $\lambda\mathsf{x}_g$-terms consists in

- taking an appropriate subsystem h that is strongly normalising, as in Lemma 32
- proving the simulation theorem in $\lambda I_{[,]}$

In the case of $\lambda\mathsf{x}_g$ and $\lambda\mathsf{x}_g^{\mathsf{SU}}$, we can in fact prove a stronger result than Theorems 30 and 34 that typed $\lambda\mathsf{x}_g$-terms are strongly normalising with respect to $\longrightarrow_{\lambda\mathsf{x}_g,\lambda\mathsf{x}_g^{\mathsf{SU}}}$, taking $\mathsf{h} := \mathsf{safe}(\mathsf{B},\mathsf{B}_3),\mathsf{min}(\mathsf{x},\mathsf{x}^{\mathsf{SU}})$.

5 Conclusion and related work

We have presented proofs of strong normalisation of typed terms using non-deterministic translations into Klop's $\lambda I_{[,]}$-calculus. The method has worked for the explicit substitution calculus in [22] extended with the rule (B_2) as well as the cut-elimination system in [27]. The proof method provides a general framework for showing strong normalisation of various reduction systems on $\lambda\mathsf{x}_g$-terms.

As regards related work, the CGPS-translation [22, 9] has been used for proving strong normalisation of calculi that correspond to proof systems with general elimination rules. (Those calculi do not have step-by-step reduction of explicit substitutions.) It aims to simulate every reduction step of the calculi by at least one β-reduction step in the λ-calculus. On the other hand, the method in this paper makes such reduction steps as few as possible, i.e., in the case of $\lambda\mathsf{x}_g$, only unsafe B-reduction steps have to be simulated by at least one β,π-reduction step in $\lambda I_{[,]}$ (cf. the remark after Definition 23). This is an essential part of our proof of strong normalisation for explicit substitution calculi.

The cut-elimination system in [27] is not intended to simulate β-reduction and was so far difficult to classify among, and relate to, other cut-elimination procedures for sequent calculus. Our proof of strong normalisation in the general framework helps to shed some light on such an exotic system. The strong normalisation proof in [27] introduces Klop's pairing constructs not only for λ-terms but also for proof terms for sequent calculus. This leads to complications, and in this sense, our approach is simpler than theirs.

Recent work by Espírito Santo and Pinto [10] has introduced some variants of intuitionistic sequent calculi (without step-by-step reduction of explicit substitutions). Reduction of those calculi is directly simulated by the explicit substitution calculus in [15], so that strong normalisation of the calculi follows from that of the calculus in [15]. On the other hand, the explicit substitution calculi we studied in this paper do not seem to be directly simulated by the calculus in [15], since it is not easy to simulate a local cut-elimination procedure in sequent calculus by an explicit substitution calculus for the usual λ-calculus.

Acknowledgements. I would like to thank the anonymous reviewers for valuable comments. I also thank Stéphane Lengrand for valuable discussions. The figures of the derivations have been drawn using Makoto Tatsuta's `proof.sty` macros.

References

1 M. Abadi, L. Cardelli, P.-L. Curien, and J.-J. Lévy. Explicit substitutions. *J. Funct. Programming*, 1(4):375–416, 1991.

2 F. Baader and T. Nipkow. *Term Rewriting and All That.* Cambridge University Press, 1998.

3 Z. Benaissa, D. Briaud, P. Lescanne, and J. Rouyer-Degli. $\lambda\upsilon$, a calculus of explicit substitutions which preserves strong normalisation. *J. Funct. Programming*, 6(5):699–722, 1996.

4 R. Bloo. *Preservation of Termination for Explicit Substitution.* PhD thesis, Eindhoven University of Technology, 1997.

5 R. Bloo and H. Geuvers. Explicit substitution: On the edge of strong normalization. *Theoret. Comput. Sci.*, 211(1-2):375–395, 1999.

6 R. Bloo and K. H. Rose. Preservation of strong normalisation in named lambda calculi with explicit substitution and garbage collection. In *Proc. of CSN'95 (Computing Science in the Netherlands)*, 62–72, 1995.

7 P.-L. Curien and H. Herbelin. The duality of computation. In *Proc. of ICFP'00*, 233–243, 2000.

8 I. L. Gørtz, S. Reuss, and M. H. Sørensen. Strong normalization from weak normalization by translation into the lambda-I-calculus. *Higher-Order and Symbolic Computation*, 16(3):253–285, 2003.

9 J. Espírito Santo, R. Matthes, and L. Pinto. Continuation-passing style and strong normalisation for intuitionistic sequent calculi. *Logical Methods in Computer Science*, 5(2), 2009.

10 J. Espírito Santo and L. Pinto. A calculus of multiary sequent terms. *ACM Trans. Comput. Log.*, 12(3):22, 2011.

11 F. Joachimski and R. Matthes. Standardization and confluence for a lambda calculus with generalized applications. In *Proc. of RTA'00*, LNCS 1833, 141–155, 2000.

12 F. Joachimski and R. Matthes. Short proofs of normalization for the simply-typed λ-calculus, permutative conversions and Gödel's T. *Arch. Math. Log.*, 42(1):59–87, 2003.

13 S. Kamin and J.-J. Lévy. Attempts for generalizing the recursive path orderings. 1980. Handwritten paper, University of Illinois.

14 D. Kesner and S. Lengrand. Resource operators for λ-calculus. *Inform. and Comput.*, 205(4):419–473, 2007.

15 D. Kesner A theory of explicit substitutions with safe and full composition. *Logical Methods in Computer Science*, 5(3), 2009.

16 K. Kikuchi and S. Lengrand. Strong normalisation of cut-elimination that simulates β-reduction. In *Proc. of FoSSaCS'08*, LNCS 4962, 380–394, 2008.

17 K. Kikuchi. Non-deterministic CPS-translation for $\lambda\mu$-calculus. 2013. Manuscript. Available at `http://www.nue.riec.tohoku.ac.jp/user/kentaro/cpslm`.

18 J. W. Klop. *Combinatory Reduction Systems*, volume 127 of *Mathematical Centre Tracts*. CWI, 1980. PhD Thesis.

19 S. Lengrand. Induction principles as the foundation of the theory of normalisation: concepts and techniques. Technical report, Université Paris 7, March 2005. Available at `http://hal.ccsd.cnrs.fr/ccsd-00004358`.

20 S. Lengrand. *Normalisation & Equivalence in Proof Theory & Type Theory*. PhD thesis, Université Paris 7 & University of St Andrews, 2006.

21 P.-A. Melliès. Typed λ-calculi with explicit substitution may not terminate. In *Proc. of TLCA'95*, LNCS 902, 328–334, 1995.

22 K. Nakazawa. An isomorphism between cut-elimination procedure and proof reduction. In *Proc. of TLCA'07*, LNCS 4583, 336–350, 2007.

23 K. Nakazawa and M. Tatsuta. Strong normalization proof with CPS-translation for second order classical natural deduction. *J. of Symbolic Logic*, 68(3):851–859, 2003. Corrigendum: vol. 68 (2003), no. 4, pp. 1415–1416.

24 K. Nakazawa and M. Tatsuta. Strong normalization of classical natural deduction with disjunctions. *Ann. Pure Appl. Logic*, 153(1–3):21–37, 2008.

25 M. Parigot. $\lambda\mu$-calculus: An algorithmic interpretation of classical natural deduction. In *Proc. of LPAR'92*, LNCS 624, 190–201, 1992.

26 J. von Plato. Natural deduction with general elimination rules. *Arch. Math. Log.*, 40(7):541–567, 2001.

27 M. H. Sørensen and P. Urzyczyn. Strong cut-elimination in sequent calculus using Klop's ι-translation and perpetual reductions. *J. of Symbolic Logic*, 73(3):919–932, 2008.

A Proof in Section 2

In this appendix we give a proof of Lemma 15 in Section 2.

First, note the following facts:

- For any λ_g-term M, $\mathsf{FV}(\mathcal{G}(M)) = \mathsf{FV}(M)$.
- For any λ-term t, $\mathsf{FV}(\mathsf{i}(t)) = \mathsf{FV}(t)$.
- For any λ-terms t and u, $\mathsf{i}(\{u/x\}t) = \{\mathsf{i}(u)/x\}\mathsf{i}(t)$.

▶ **Lemma 15.** *For any λ_g-term M, there exists a $\lambda I_{[,]}$-term T such that $M \mathrel{\mathcal{H}} T$ and $\mathsf{i}(\mathcal{G}(M)) \longrightarrow^*_{\beta,\pi} T$.*

Proof. By induction on M. The case where M is a variable is straightforward. We consider the remaining two cases.

- For $\lambda x.M$, we have

$$\mathsf{i}(\mathcal{G}(\lambda x.M)) = \mathsf{i}(\lambda x.\mathcal{G}(M)) = \begin{cases} \lambda x.\mathsf{i}(\mathcal{G}(M)) & \text{if } x \in \mathsf{FV}(\mathcal{G}(M)) \\ \lambda x.[\mathsf{i}(\mathcal{G}(M)), x] & \text{if } x \notin \mathsf{FV}(\mathcal{G}(M)) \end{cases}$$

  By the induction hypothesis, there is a $\lambda I_{[,]}$-term T such that $M \mathrel{\mathcal{H}} T$ and $\mathsf{i}(\mathcal{G}(M)) \longrightarrow^*_{\beta,\pi} T$.
  - If $x \in \mathsf{FV}(\mathcal{G}(M))$ then $\mathsf{i}(\mathcal{G}(\lambda x.M)) = \lambda x.\mathsf{i}(\mathcal{G}(M)) \longrightarrow^*_{\beta,\pi} \lambda x.T$, and since $\mathsf{FV}(\mathcal{G}(M)) = \mathsf{FV}(\mathsf{i}(\mathcal{G}(M))) = \mathsf{FV}(T)$, we have $x \in \mathsf{FV}(T)$. From $M \mathrel{\mathcal{H}} T$, we have $\lambda x.M \mathrel{\mathcal{H}} \lambda x.T$.
  - If $x \notin \mathsf{FV}(\mathcal{G}(M))$ then $\mathsf{i}(\mathcal{G}(\lambda x.M)) = \lambda x.[\mathsf{i}(\mathcal{G}(M)), x] \longrightarrow^*_{\beta,\pi} \lambda x.[T, x]$. From $M \mathrel{\mathcal{H}} T$, we have $M \mathrel{\mathcal{H}} [T, x]$, and hence $\lambda x.M \mathrel{\mathcal{H}} \lambda x.[T, x]$.
- For $M[N, y.P]$, we have

$$\mathsf{i}(\mathcal{G}(M[N, y.P])) = \begin{cases} \{\mathsf{i}(\mathcal{G}(M))\,\mathsf{i}(\mathcal{G}(N))/y\}\mathsf{i}(\mathcal{G}(P)) & \text{if } y \in \mathsf{FV}(P) \\ (\lambda y.[\mathsf{i}(\mathcal{G}(P)), y])(\mathsf{i}(\mathcal{G}(M))\,\mathsf{i}(\mathcal{G}(N))) & \text{if } y \notin \mathsf{FV}(P) \end{cases}$$

  By the induction hypothesis, there are $\lambda I_{[,]}$-terms T, U and S such that (a) $M \mathrel{\mathcal{H}} T$ and $\mathsf{i}(\mathcal{G}(M)) \longrightarrow^*_{\beta,\pi} T$, (b) $N \mathrel{\mathcal{H}} U$ and $\mathsf{i}(\mathcal{G}(N)) \longrightarrow^*_{\beta,\pi} U$, and (c) $P \mathrel{\mathcal{H}} S$ and $\mathsf{i}(\mathcal{G}(P)) \longrightarrow^*_{\beta,\pi} S$.
  - If $y \in \mathsf{FV}(P)$ then $\mathsf{i}(\mathcal{G}(M[N, y.P])) = \{\mathsf{i}(\mathcal{G}(M))\,\mathsf{i}(\mathcal{G}(N))/y\}\mathsf{i}(\mathcal{G}(P)) \longrightarrow^*_{\beta,\pi} \{T\,U/y\}S$. Since $\mathsf{FV}(P) = \mathsf{FV}(\mathsf{i}(\mathcal{G}(P))) = \mathsf{FV}(S)$, we have $y \in \mathsf{FV}(S)$. Hence, from (a), (b) and (c), we have $M[N, y.P] \mathrel{\mathcal{H}} \{T\,U/y\}S$.
  - If $y \notin \mathsf{FV}(P)$ then $\mathsf{i}(\mathcal{G}(M[N, y.P])) = (\lambda y.[\mathsf{i}(\mathcal{G}(P)), y])(\mathsf{i}(\mathcal{G}(M))\,\mathsf{i}(\mathcal{G}(N))) \longrightarrow^*_{\beta,\pi} (\lambda y.[S, y])(T\,U) \longrightarrow_\beta \{T\,U/y\}[S, y]$. From (c), we have $P \mathrel{\mathcal{H}} [S, y]$ and hence $M[N, y.P] \mathrel{\mathcal{H}} \{T\,U/y\}[S, y]$.

◀

B Proofs in Section 3

In this appendix we give proofs of Lemmas 26 and 29, and Theorem 27 in Section 3.

In the proof below we use the following fact:

- If N is a proper subterm of M then $w(M) > w(N)$ and hence $\mathsf{c}^{w(M)} > \mathsf{c}^{w(N)}$.

▶ **Lemma 26.** *If $M \longrightarrow_{\mathsf{safeB},\mathsf{minx}} M'$ then $\overline{M} >_{\mathsf{lpo}} \overline{M'}$. Hence, $\longrightarrow_{\mathsf{safeB},\mathsf{minx}}$ is strongly normalising.*

Proof. By induction on the derivation of the reduction step. First we consider the cases where the reduction takes place at the root. If the reduction step is safe, i.e. if the redex itself is in $\mathsf{SN}^{\lambda\mathsf{x}_g}$, then $\overline{M} \equiv \mathsf{c}^{w(M)} >_{\mathsf{lpo}} \mathsf{c}^{w(M')} \equiv \overline{M'}$. So let the reduction step be $\longrightarrow_{\mathsf{minx}}$ where the redex is not in $\mathsf{SN}^{\lambda\mathsf{x}_g}$.

(1) $\langle M/x\rangle y \longrightarrow_{\mathsf{minx}} y \quad (x \not\equiv y)$

$$\begin{array}{lll} \text{LHS}: & \overline{\langle M/x\rangle y} & = \mathsf{sub}(\overline{M}, \overline{y}) \\ \text{RHS}: & \overline{y} & = \overline{y} \end{array}$$

(2) $\langle M/x\rangle x \longrightarrow_{\mathsf{minx}} M$

$$\begin{array}{lll} \text{LHS}: & \overline{\langle M/x\rangle x} & = \mathsf{sub}(\overline{M}, \overline{x}) \\ \text{RHS}: & \overline{M} & = \overline{M} \end{array}$$

(3) $\langle M/x\rangle(\lambda y.N) \longrightarrow_{\mathsf{minx}} \lambda y.\langle M/x\rangle N$

$$\begin{array}{lll} \text{LHS}: & \overline{\langle M/x\rangle(\lambda y.N)} & = \mathsf{sub}(\overline{M}, \overline{\lambda y.N}) \\ & & = \mathsf{sub}(\overline{M}, \mathsf{c}^{w(\lambda y.N)}) \\ \text{RHS}: & \overline{\lambda y.\langle M/x\rangle N} & = \mathsf{abs}(\overline{\langle M/x\rangle N}) \\ & & = \mathsf{abs}(\mathsf{sub}(\overline{M}, \overline{N})) \\ & & = \mathsf{abs}(\mathsf{sub}(\overline{M}, \mathsf{c}^{w(N)})) \end{array}$$

(4) $\langle M/x\rangle(y[N, z.P]) \longrightarrow_{\mathsf{minx}} y[\langle M/x\rangle N, z.\langle M/x\rangle P] \quad (x \not\equiv y)$

$$\begin{array}{lll} \text{LHS}: & \overline{\langle M/x\rangle(y[N, z.P])} & = \mathsf{sub}(\overline{M}, \overline{y[N, z.P]}) \\ & & = \mathsf{sub}(\overline{M}, \mathsf{c}^{w(y[N,z.P])}) \\ \text{RHS}: & \overline{y[\langle M/x\rangle N, z.\langle M/x\rangle P]} & \leq \mathsf{gapp}(\overline{y}, \overline{\langle M/x\rangle N}, \overline{\langle M/x\rangle P}) \\ & & \leq \mathsf{gapp}(\overline{y}, \mathsf{sub}(\overline{M}, \overline{N}), \mathsf{sub}(\overline{M}, \overline{P})) \\ & & = \mathsf{gapp}(\mathsf{c}^{w(y)}, \mathsf{sub}(\overline{M}, \mathsf{c}^{w(N)}), \mathsf{sub}(\overline{M}, \mathsf{c}^{w(P)})) \end{array}$$

where $\leq$ is used for $= \cup <_{\mathsf{lpo}}$ to deal with the cases where some of the subterms of RHS are already in $\mathsf{SN}^{\lambda x_g}$, in which cases those subterms M are encoded as $\mathsf{c}^{w(M)}$.

(5) $\langle M/x\rangle(x[N, z.P]) \longrightarrow_{\mathsf{minx}} M[\langle M/x\rangle N, z.\langle M/x\rangle P] \quad (x \in \mathsf{FV}([N, z.P]))$

$$\begin{array}{lll} \text{LHS}: & \overline{\langle M/x\rangle(x[N, z.P])} & = \mathsf{sub}(\overline{M}, \overline{x[N, y.P]}) \\ & & = \mathsf{sub}(\overline{M}, \mathsf{c}^{w(x[N,y.P])}) \\ \text{RHS}: & \overline{M[\langle M/x\rangle N, z.\langle M/x\rangle P]} & \leq \mathsf{gapp}(\overline{M}, \overline{\langle M/x\rangle N}, \overline{\langle M/x\rangle P}) \\ & & \leq \mathsf{gapp}(\overline{M}, \mathsf{sub}(\overline{M}, \overline{N}), \mathsf{sub}(\overline{M}, \overline{P})) \\ & & = \mathsf{gapp}(\overline{M}, \mathsf{sub}(\overline{M}, \mathsf{c}^{w(N)}), \mathsf{sub}(\overline{M}, \mathsf{c}^{w(P)}))) \end{array}$$

(6) $\langle M/x\rangle(Q[N, z.P]) \longrightarrow_{\mathsf{minx}} (\langle M/x\rangle Q)[\langle M/x\rangle N, z.\langle M/x\rangle P] \quad (Q \text{ is not a variable})$

$$\begin{array}{lll} \text{LHS}: & \overline{\langle M/x\rangle(Q[N, z.P])} & = \mathsf{sub}(\overline{M}, \overline{Q[N, y.P]}) \\ & & = \mathsf{sub}(\overline{M}, \mathsf{c}^{w(Q[N,y.P])}) \\ \text{RHS}: & \overline{(\langle M/x\rangle Q)[\langle M/x\rangle N, z.\langle M/x\rangle P]} & \leq \mathsf{gapp}(\overline{\langle M/x\rangle Q}, \overline{\langle M/x\rangle N}, \overline{\langle M/x\rangle P}) \\ & & \leq \mathsf{gapp}(\mathsf{sub}(\overline{M}, \overline{Q}), \mathsf{sub}(\overline{M}, \overline{N}), \mathsf{sub}(\overline{M}, \overline{P})) \\ & & = \mathsf{gapp}(\mathsf{sub}(\overline{M}, \mathsf{c}^{w(Q)}), \mathsf{sub}(\overline{M}, \mathsf{c}^{w(N)}), \mathsf{sub}(\overline{M}, \mathsf{c}^{w(P)})) \end{array}$$

(7) $\langle M/x\rangle(x[N, z.P]) \longrightarrow_{\mathsf{minx}} M[N, z.P] \quad (x \notin \mathsf{FV}([N, z.P]))$

$$\begin{array}{lll} \text{LHS}: & \overline{\langle M/x\rangle(x[N, z.P])} & = \mathsf{sub}(\overline{M}, \overline{x[N, z.P]}) \\ & & = \mathsf{sub}(\overline{M}, \mathsf{c}^{w(x[N,z.P])}) \\ \text{RHS}: & \overline{M[N, z.P]} & \leq \mathsf{gapp}(\overline{M}, \overline{N}, \overline{P}) \\ & & = \mathsf{gapp}(\overline{M}, \mathsf{c}^{w(N)}, \mathsf{c}^{w(P)}) \end{array}$$

(Pi) $M[N, z.P][N', z'.P'] \longrightarrow_{\mathsf{min}\times} M[N, z.P[N', z'.P']]$

$$\begin{array}{lll} \text{LHS}: & \overline{M[N, z.P][N', z'.P']} & = \mathsf{gapp}(\overline{M[N, z.P]}, \overline{N'}, \overline{P'}) \\ & & = \mathsf{gapp}(\mathsf{c}^{w(M[N,z.P])}, \overline{N'}, \overline{P'}) \\ \text{RHS}: & \overline{M[N, z.P[N', z'.P']]} & \leq \mathsf{gapp}(\overline{M}, \overline{N}, \overline{P[N', z'.P']}) \\ & & \leq \mathsf{gapp}(\overline{M}, \overline{N}, \mathsf{gapp}(\overline{P}, \overline{N'}, \overline{P'})) \\ & & = \mathsf{gapp}(\mathsf{c}^{w(M)}, \mathsf{c}^{w(N)}, \mathsf{gapp}(\mathsf{c}^{w(P)}, \overline{N'}, \overline{P'})) \end{array}$$

The cases where the reduction is not at the root are easily proved by the induction hypothesis, since $>_{\mathsf{lpo}}$ is context-closed. ◀

▶ **Theorem 27** (Simulation in $\lambda I_{[,]}$)**.** *Suppose $M \mathrel{\mathcal{H}} T$.*

1. *If $M \longrightarrow_{minB} N$ and the reduction step is unsafe then there exists U such that $N \mathrel{\mathcal{H}} U$ and $T \longrightarrow^{+}_{\beta,\pi} U$.*
2. *If $M \longrightarrow_{minB} N$ and the reduction step is safe then there exists U such that $N \mathrel{\mathcal{H}} U$ and $T \longrightarrow^{*}_{\beta,\pi} U$.*
3. *If $M \longrightarrow_{minx} N$ then $N \mathrel{\mathcal{H}} T$.*

Proof. By induction on the derivation of $M \mathrel{\mathcal{H}} T$. Here we consider the cases where the reduction takes place at the root and those where the derivation ends with the rule for explicit substitution. (The other cases are proved in the same way as in the proof of Theorem 14.)

First we inspect the case where one of (B_1), (B_2) and (Pi) takes place at the root. Note that, by minimality, M, N and P (in the rules below) are in $\mathsf{SN}^{\lambda x_g}$.

(B_1) $(\lambda y.M)[N, z.P] \longrightarrow_{\mathsf{minB}} \langle N/y\rangle\langle M/z\rangle P$. In this case, the derivation has the form

$$\dfrac{\dfrac{\dfrac{M \mathrel{\mathcal{H}} T}{\lambda y.M \mathrel{\mathcal{H}} \lambda y.T}\; y \in \mathsf{FV}(T)}{\lambda y.M \mathrel{\mathcal{H}} [\lambda y.T, \overrightarrow{R}]} \quad N \mathrel{\mathcal{H}} U \quad P \mathrel{\mathcal{H}} S}{(\lambda y.M)[N, z.P] \mathrel{\mathcal{H}} \{[\lambda y.T, \overrightarrow{R}]U/z\}S}\; z \in \mathsf{FV}(S)$$

Then we have

$$\dfrac{N \mathrel{\mathcal{H}} U \quad \dfrac{\dfrac{M \mathrel{\mathcal{H}} T}{M \mathrel{\mathcal{H}} [T, \overrightarrow{R}]} \quad P \mathrel{\mathcal{H}} S}{\langle M/z\rangle P \mathrel{\mathcal{H}} \{[T, \overrightarrow{R}]/z\}S}}{\langle N/y\rangle\langle M/z\rangle P \mathrel{\mathcal{H}} \{U/y\}\{[T, \overrightarrow{R}]/z\}S \equiv \{[\{U/y\}T, \overrightarrow{R}]/z\}S}$$

Since $z \in \mathsf{FV}(S)$, we have $\{[\lambda y.T, \overrightarrow{R}]U/z\}S \longrightarrow^{+}_{\beta,\pi} \{[\{U/y\}T, \overrightarrow{R}]/z\}S$ as required.

(B_2) $(\lambda y.M)[N, z.P] \longrightarrow_{\mathsf{minB}} \langle\langle N/y\rangle M/z\rangle P$. In this case, the derivation has the same form as the case (B_1). Then we have

$$\dfrac{\dfrac{\dfrac{N \mathrel{\mathcal{H}} U \quad M \mathrel{\mathcal{H}} T}{\langle N/y\rangle M \mathrel{\mathcal{H}} \{U/y\}T}}{\langle N/y\rangle M \mathrel{\mathcal{H}} [\{U/y\}T, \overrightarrow{R}]} \quad P \mathrel{\mathcal{H}} S}{\langle\langle N/y\rangle M/z\rangle P \mathrel{\mathcal{H}} \{[\{U/y\}T, \overrightarrow{R}]/z\}S}$$

Again, since $z \in \mathsf{FV}(S)$, we have $\{[\lambda y.T, \overrightarrow{R}]U/z\}S \longrightarrow^{+}_{\beta,\pi} \{[\{U/y\}T, \overrightarrow{R}]/z\}S$ as required.

(Pi) $M[N, z.P][N', z'.P'] \longrightarrow_{\mathsf{min}\times} M[N, z.P[N', z'.P']]$. This case is proved in the same way as the case (π_g) in the proof of Theorem 14.

Next we consider the case where the last applied rule of the derivation is

$$\frac{M \mathrel{\mathcal{H}} T \quad N \mathrel{\mathcal{H}} U}{\langle M/x\rangle N \mathrel{\mathcal{H}} \{T/x\}U}\ x \in \mathsf{FV}(U) \vee M \in \mathsf{SN}^{\lambda \mathsf{x}_g}$$

If the reduction takes place within M or N, we apply the induction hypothesis, remembering that reduction in $\lambda I_{[,]}$ preserves free variables (Lemma 6), so the side-condition remains satisfied. Moreover, an unsafe B-reduction in M is simulated by at least one reduction step from T. (Indeed, since the B-reduction is unsafe, we know that $M \notin \mathsf{SN}^{\lambda \mathsf{x}_g}$ and hence we must have $x \in \mathsf{FV}(U)$.) The simulating reduction step from T is therefore preserved in the reduction of $\{T/x\}U$. This is the precise point where the distinction between safe and unsafe reductions plays its role.

Otherwise, we have a (minimal) root reduction and the case analysis below inspects some of the rules. Note that, by minimality, both M and N (in the rule above) are in $\mathsf{SN}^{\lambda \mathsf{x}_g}$.

(1) $\langle M/x\rangle y \longrightarrow_{\mathsf{minx}} y$ $(x \not\equiv y)$. In this case, the derivation has the form

$$\frac{M \mathrel{\mathcal{H}} T \quad \dfrac{\overline{y \mathrel{\mathcal{H}} y}}{y \mathrel{\mathcal{H}} [y, \overrightarrow{R}]}}{\langle M/x\rangle y \mathrel{\mathcal{H}} \{T/x\}[y, \overrightarrow{R}] \equiv [y, \{T/x\}\overrightarrow{R}]}$$

Then we have

$$\frac{\overline{y \mathrel{\mathcal{H}} y}}{y \mathrel{\mathcal{H}} [y, \{T/x\}\overrightarrow{R}]}$$

(2) $\langle M/x\rangle x \longrightarrow_{\mathsf{minx}} M$. In this case, the derivation has the form

$$\frac{M \mathrel{\mathcal{H}} T \quad \dfrac{\overline{x \mathrel{\mathcal{H}} x}}{x \mathrel{\mathcal{H}} [x, \overrightarrow{R}]}}{\langle M/x\rangle x \mathrel{\mathcal{H}} \{T/x\}[x, \overrightarrow{R}] \equiv [T, \{T/x\}\overrightarrow{R}]}$$

Then we have

$$\frac{M \mathrel{\mathcal{H}} T}{M \mathrel{\mathcal{H}} [T, \{T/x\}\overrightarrow{R}]}$$

(3) $\langle M/x\rangle(\lambda y.N) \longrightarrow_{\mathsf{minx}} \lambda y.\langle M/x\rangle N$. In this case, the derivation has the form

$$\frac{M \mathrel{\mathcal{H}} T \quad \dfrac{\dfrac{N \mathrel{\mathcal{H}} U}{\lambda y.N \mathrel{\mathcal{H}} \lambda y.U}\ y \in \mathsf{FV}(U)}{\lambda y.N \mathrel{\mathcal{H}} [\lambda y.U, \overrightarrow{R}]}}{\langle M/x\rangle(\lambda y.N) \mathrel{\mathcal{H}} \{T/x\}[\lambda y.U, \overrightarrow{R}] \equiv [\lambda y.\{T/x\}U, \{T/x\}\overrightarrow{R}]}$$

Then we have

$$\frac{\dfrac{\dfrac{M \mathrel{\mathcal{H}} T \quad N \mathrel{\mathcal{H}} U}{\langle M/x\rangle N \mathrel{\mathcal{H}} \{T/x\}U}}{\lambda y.\langle M/x\rangle N \mathrel{\mathcal{H}} \lambda y.\{T/x\}U}\ y \in \mathsf{FV}(\{T/x\}U)}{\lambda y.\langle M/x\rangle N \mathrel{\mathcal{H}} [\lambda y.\{T/x\}U, \{T/x\}\overrightarrow{R}]}$$

(6) $\langle M/x\rangle(Q[N, z.P]) \longrightarrow_{\mathsf{minx}} (\langle M/x\rangle Q)[\langle M/x\rangle N, z.\langle M/x\rangle P]$ (Q is not a variable). In this case, the derivation has the form

$$\frac{M \mathrel{\mathcal{H}} T \quad \dfrac{\dfrac{Q \mathrel{\mathcal{H}} T' \quad N \mathrel{\mathcal{H}} U \quad P \mathrel{\mathcal{H}} S}{Q[N, z.P] \mathrel{\mathcal{H}} \{T'U/z\}S}\ z \in \mathsf{FV}(S)}{Q[N, z.P] \mathrel{\mathcal{H}} [\{T'U/z\}S, \overrightarrow{R}]}}{\langle M/x\rangle(Q[N, z.P]) \mathrel{\mathcal{H}} \{T/x\}[\{T'U/z\}S, \overrightarrow{R}]}$$

Then we have

$$\dfrac{\dfrac{\dfrac{M\ \mathcal{H}\ T \quad Q\ \mathcal{H}\ T'}{\langle M/x\rangle Q\ \mathcal{H}\ \{T/x\}T'} \quad \dfrac{M\ \mathcal{H}\ T \quad N\ \mathcal{H}\ U}{\langle M/x\rangle N\ \mathcal{H}\ \{T/x\}U} \quad \dfrac{M\ \mathcal{H}\ T \quad P\ \mathcal{H}\ S}{\langle M/x\rangle P\ \mathcal{H}\ \{T/x\}S}}{(\langle M/x\rangle Q)[\langle M/x\rangle N, z.\langle M/x\rangle P]\ \mathcal{H}\ \{\{T/x\}T'\{T/x\}U/z\}\{T/x\}S}\ z \in \mathsf{FV}(\{T/x\}S)}{(\langle M/x\rangle Q)[\langle M/x\rangle N, z.\langle M/x\rangle P]\ \mathcal{H}\ [\{\{T/x\}T'\{T/x\}U/z\}\{T/x\}S, \{T/x\}\overrightarrow{R}]}$$
$$\Vert\!\Vert$$
$$(\langle M/x\rangle Q)[\langle M/x\rangle N, z.\langle M/x\rangle P]\ \mathcal{H}\ \{T/x\}[\{T'U/z\}S, \overrightarrow{R}]$$

(7) $\langle M/x\rangle(x[N, z.P]) \longrightarrow_{\mathsf{minx}} M[N, z.P]$ $(x \notin \mathsf{FV}([N, z.P]))$. In this case, the derivation has the form

$$\dfrac{\dfrac{M\ \mathcal{H}\ T \qquad \dfrac{\dfrac{x\ \mathcal{H}\ T' \quad N\ \mathcal{H}\ U \quad P\ \mathcal{H}\ S}{x[N, z.P]\ \mathcal{H}\ \{T'U/z\}S}\ z \in \mathsf{FV}(S)}{x[N, z.P]\ \mathcal{H}\ [\{T'U/z\}S, \overrightarrow{R}]}}{\langle M/x\rangle(x[N, z.P])\ \mathcal{H}\ \{T/x\}[\{T'U/z\}S, \overrightarrow{R}]}$$

where $T' \equiv [x, \overrightarrow{R'}]$. Then we have

$$\dfrac{\dfrac{\dfrac{M\ \mathcal{H}\ T}{M\ \mathcal{H}\ [T, \{T/x\}\overrightarrow{R'}]}}{M\ \mathcal{H}\ \{T/x\}T'} \quad \dfrac{N\ \mathcal{H}\ U}{N\ \mathcal{H}\ \{T/x\}U}\ \text{Lemma 11 (0c)} \quad \dfrac{P\ \mathcal{H}\ S}{P\ \mathcal{H}\ \{T/x\}S}\ \text{Lemma 11 (0c)}}{\dfrac{M[N, z.P]\ \mathcal{H}\ \{\{T/x\}T'\{T/x\}U/z\}\{T/x\}S}{M[N, z.P]\ \mathcal{H}\ [\{\{T/x\}T'\{T/x\}U/z\}\{T/x\}S, \{T/x\}\overrightarrow{R}]}}\ z \in \mathsf{FV}(\{T/x\}S)$$
$$\Vert\!\Vert$$
$$M[N, z.P]\ \mathcal{H}\ \{T/x\}[\{T'U/z\}S, \overrightarrow{R}]$$

◀

▶ **Lemma 29.** *For any λx_g-term M, there exists a $\lambda I_{[,]}$-term T such that $M\ \mathcal{H}\ T$ and $\mathsf{i}(\mathcal{G}(M)) \longrightarrow^*_{\beta,\pi} T$.*

Proof. By induction on M. Here we consider the case of explicit substitution. (The other cases are proved in the same way as in the proof of Lemma 15.) Then we have

$$\mathsf{i}(\mathcal{G}(\langle M/x\rangle N)) = \begin{cases} \{\mathsf{i}(\mathcal{G}(M))/x\}\mathsf{i}(\mathcal{G}(N)) & \text{if } x \in \mathsf{FV}(N) \\ (\lambda x.[\mathsf{i}(\mathcal{G}(N)), x])\,\mathsf{i}(\mathcal{G}(M)) & \text{if } x \notin \mathsf{FV}(N) \end{cases}$$

By the induction hypothesis, there are $\lambda I_{[,]}$-terms T and U such that (a) $M\ \mathcal{H}\ T$ and $\mathsf{i}(\mathcal{G}(M)) \longrightarrow^*_{\beta,\pi} T$, and (b) $N\ \mathcal{H}\ U$ and $\mathsf{i}(\mathcal{G}(N)) \longrightarrow^*_{\beta,\pi} U$.

- If $x \in \mathsf{FV}(N)$ then $\mathsf{i}(\mathcal{G}(\langle M/x\rangle N)) = \{\mathsf{i}(\mathcal{G}(M))/x\}\mathsf{i}(\mathcal{G}(N)) \longrightarrow^*_{\beta,\pi} \{T/x\}U$. Since $\mathsf{FV}(N) = \mathsf{FV}(\mathsf{i}(\mathcal{G}(N))) = \mathsf{FV}(U)$, we have $x \in \mathsf{FV}(U)$. Hence, from (a) and (b), we have $\langle M/x\rangle N\ \mathcal{H}\ \{T/x\}U$.
- If $x \notin \mathsf{FV}(N)$ then $\mathsf{i}(\mathcal{G}(\langle M/x\rangle N)) = (\lambda x.[\mathsf{i}(\mathcal{G}(N)), x])\,\mathsf{i}(\mathcal{G}(M)) \longrightarrow^*_{\beta,\pi} (\lambda x.[U, x])T \longrightarrow_\beta \{T/x\}[U, x]$. From (b), we have $N\ \mathcal{H}\ [U, x]$ and hence $\langle M/x\rangle N\ \mathcal{H}\ \{T/x\}[U, x]$.

◀