Technical Report 25/96, Department of Computing Science, University of Karlsruhe

Separating Weakening and Contraction in a Linear Lambda Calculus (Unabridged)

John Maraist*

Abstract. *We present a separated-linear lambda calculus based on a refinement of linear logic which allows separate control of weakening and contraction. The calculus satisfies subject reduction and confluence, has a straightforward notion of standard evaluation, and inherits previous results on the relationship of Girard's two translations from minimal intuitionistic logic to linear logic with call-by-name and call-by-value. We construct a hybrid translation from Girard's two which is sound and complete for mapping types, reduction sequences and standard evaluation sequences from call-by-need into separated-linear λ, a more satisfying treatment of call-by-need than in previous work, which now allows a contrasting of all three reduction strategies in the manner (for example) that the CPS translations allow for call-by-name and call-by-value.*

ONE fundamental application of the continuation-passing transformations is to explain the different parameter-passing mechanisms of call-by-name and call-by-value by different translations from each respective λ-calculus into the continuation-passing calculus [21, 22]. Each transformation makes the control operations of the respective mechanism explicit in the transformed term, so that reduction of the transformed term by either mechanism produces equivalent results. In this paper, we will also compare different calling mechanisms by mapping them into a common system, but rather than focusing on an explicit flow of control such as through the continuation-passing style, we will contrast the mechanisms in terms of the substructural operations of the lambda calculus using *linear* systems, where the operations of weakening and contraction are allowed only in conjunction with a new modal connective. Moreover, rather than considering linear systems with a single intuitionistic mode, as in previous approaches to this problem, we will construct as a target of the translations a *separated*-linear lambda calculus, where the two key substructural operations of weakening and contraction are enabled by distinct modal connectives.

Typed lambda calculi generally have a Curry-Howard correspondence [11, 14]: a close relationship between their type systems and formal logical frameworks, usually some variety of minimal intuitionistic logic. In such logics, structural inference rules play an important, if often overlooked, role: weakening allows assumptions to be discarded, while contraction allows duplication. Moreover, these rules are the *only* facility for duplicating and discarding assumptions. So at the level of the calculi, the corresponding typing rules are the mechanisms by which we introduce the copying or discarding of terms. A term is discarded when it is substituted for a variable which appears only on the left side of a typing judgment, which can occur only as the result of applying a weakening rule; a term is duplicated when it is substituted for a variable which appears more than once on the right side of a typing judgment, which can occur only by a contraction rule.

In the three substructural lambda calculi λI [10], λA and λL [15], the use of (respectively) weakening, contraction and both are simply banned. These systems are not sufficient here: while we do want to make the use of such rules explicit, we do not want to prohibit them altogether. Instead, we will take as target calculi systems whose type systems are related to logics where the use of the structural rules must be explicitly enabled.

In previous work [20], we explored translations into a system based on the linear logic of Girard [12]. In linear logic the ability to weaken or contract is expressed in a formula by a new connective !. Linear implication requires the precedent to be used exactly once in the proof of the consequent, and is written with a new connective $\multimap$. Girard identified an intuitionistic subset of linear logic which is of particular interest to us here, so henceforward when we write "linear logic" we will mean this fragment, and will refer to the entire system of Girard as "full linear logic." Girard described two translations of intuitionistic logic into linear logic, which center around the question of the appropriate image in linear logic for the proposition $A \to B$: under his *standard* translation $\circ$, we map intuitionistic implication to $(!A^\circ) \multimap (B^\circ)$, while under his *steadfast* translation $*$ we map to $!(A^* \multimap B^*)$. Girard introduced the terms to relate formulas in the logics; the extensions of the translations to encompass term reduction became clear later: both translations preserve typability of terms; the standard (respectively steadfast) translation is sound and complete (sound) for mapping general reduction sequences from call-by-name (call-by-value) [17, 20]; both

*Institut f. Programmstrukturen und Datenorganisation, Uni. Karlsruhe (TH), Postfach 6980, D-76128 Karlsruhe, Germany. Email, maraist@ira.uka.de; web URL, http://wwwipd.ira.uka.de/~maraist/Papers.

are sound and complete for mapping standard reduction sequences [18].

The steadfast translation can also address call-by-need. Call-by-need as we presented it with Ariola, Felleisen, Odersky and Wadler [2] is a more suitable basis for the analysis of lazy functional languages than call-by-name. Although call-by-need uses a different notion of reduction than call-by-name, their observational equivalence theories are the same for convergence both to constants and to function-based closures. Unlike call-by-name — but like call-by-value and implementations of lazy functional languages such as Haskell — call-by-need also has the property that reduction will only duplicate terms which have been reduced to values.

Call-by-need cannot be straightforwardly mapped into the linear lambda calculus. The usual transformation would not be sound for reduction because call-by-need allows any term to be discarded once it is clear that the term will not be used; in the transformation into the linear calculus, only values are marked as discardable: hence we previously used the *affine* calculus as the target of the translation from call-by-need [20]. In affine systems, only copying is restricted; any term may be discarded. This arrangement has two major drawbacks: first, like call-by-value, the translation of call-by-need reduction sequences to affine reduction sequences is not complete. This restriction undermines the value of the translation as a tool for reasoning about programs, since reduction of the transformed term may suggest transformation not valid in the original system. A second disadvantage is the fact that we must use separate systems as the target of the translations. Much of the insight into call-by-name and call-by-value provided by the CPS or Girard translations arises from the fact that the distinct calculi are mapped into a single system; by using a separate target for call-by-need the insights are somewhat muddled.

In this paper we consider the separate control of weakening and contraction. We present a separated-linear lambda calculus which has two different connectives $!^w$ and $!^c$ for weakening and contraction, respectively, rather than a single ! operator enabling both together. The adaptation of Girard's two transformations into this system is simple, and not surprisingly the adapted transformations enjoy the same properties as the respective transformations into the unitary linear system. What is more insightful is the treatment the separated system allows of call-by-need.

Call-by-name and call-by-value calculi translate as well as they do into the linear calculus in part because the opportunities for discarding and duplication in these systems are coincident. In call-by-name, discarding and duplication is allowed for — but only for — any function argument; in call-by-value, for any value. Call-by-need, on the other hand, allows any function argument to be discarded, but any value to be duplicated: in the separated linear calculus we can separately describe enabling the substructural operations as suggested by this intuition. We need not resort to an affine system, since the separated calculus includes affine-style reduction. Thus we present a hybrid translation $\circledast$ from call-by-need into the separated calculus, taking intuitionistic implication $A \to B$ to the separated-linear type $!^c((!^w A^{\circledast}) \multimap B^{\circledast})$. This new translation is sound and complete for all typing judgments, reduction and standard reduction.

Related Work. Several approaches to the integration of concepts from linear logic and the original Girard translations into the lambda calculus have been presented, including (but not exhaustively) comments in Girard's original paper on linear logic [12]; the work of Abramsky [1], Benton, Bierman, de Paiva and Hyland [6, 7, 8, 9], Plotkin and Barber [4], della Rocca and Roversi [23], and Wadler [24, 25]; and our previous work [18, 20], which is the specific framework we extend here; all of these approaches have used a single mode for enabling both weakening and contraction. Jacobs [16] has addressed the issue of separate connectives for weakening and contraction, but for full linear logic and only in the model theory. To our knowledge, there have been no previous attempts to allow individual control over weakening and contraction, nor to construct the translation we call $\circledast$ for call-by-need.

Based on his earlier work on models for the substructural lambda calculi [15], Jacobs identified a category of models for a variant of linear logic with separate exponential connectives for each of weakening and contraction which generalizes affine and relevance logic as special cases [16]. Jacobs limited the scope of his study to the model theory, giving only a brief sketch of a possible syntax for the logic and not considering its relation to intuitionistic systems. It would be fair to characterize the calculus we present here as a syntax for an intuitionistic fragment of Jacobs' system.

This paper is organized as follows. We begin with a review of the relevant background. We consider three calculi which make no restrictions on the use of weakening and contraction in Section 1.1: the call-by-name, call-by-value and call-by-need systems. We are building on previous work on calculi with a single mode for enabling structural operations, and specifically on the linear and affine lambda calculi which we review in Section 1.2. The core of the present work is Section 2, where we introduce the separated-linear calculus and the three translations into it. Finally we conclude in Section 3 with a discussion of various applications and extensions. Following the primary material, we present detailed discussions of marked separated reduction and the standard reduction result in two appendices.

1 Background

1.1 Intuitionistic calculi

Figure 1 presents the details of the call-by-name and call-by-value systems, and Figure 2 presents the details of the call-by-need calculus. The properties of the first two are well-known [10, 21]; details of call-by-need are less commonly known, but are available in previous work [2, 3, 18, 19]. Here and subsequently, we let L, M, N range over terms with a subset of values over

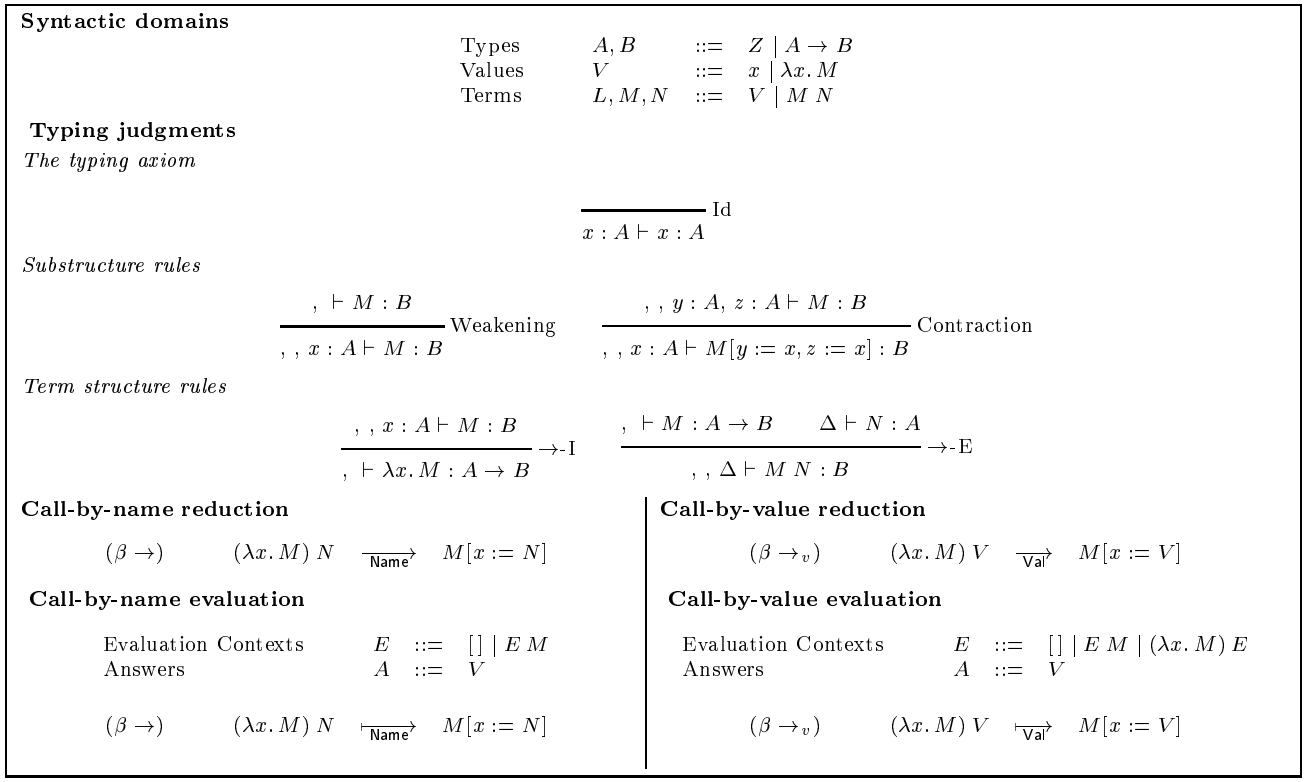


Figure 1: The call-by-name and call-by-value calculi.

which V, W range. We assume the existence of certain base types, ranged over by Z, which we leave unspecified. Types, over which A, B range, are either the base type or a function type. Values are plain variables and lambda abstractions. In Name and Val, terms can be a value or an application; in Need terms may also take the form of a let-binding, which describes the sharing in a term. We omit the name of the reduction relation from under symbols when no confusion will result. All reduction is taken to be compatibly closed under arbitrary contexts, with $\to$ denoting single steps and $\twoheadrightarrow$ for zero or more steps. We distinguish the unique evaluation sequence of a term to an answer from the general compatibly closed reduction relation by writing the former as $\mapsto$ rather than the $\to$ of the latter. The properties of these systems are investigated elsewhere:

Proposition 1.1 *Properties of call-by-name [5, 10]:*
- Name *satisfies subject reduction.*
- Name *is confluent.*
- *If* $M \xrightarrow[\mathsf{Name}]{}\!\!\!\!\twoheadrightarrow A$ *then there exists some call-by-name answer* A_0 *such that* $M \mapsto_{\mathsf{Name}}\!\!\!\!\twoheadrightarrow A_0$.

Proposition 1.2 *Properties of call-by-value [21]:*
- Val *satisfies subject reduction.*
- Val *is confluent.*
- *If* $M \xrightarrow[\mathsf{Val}]{}\!\!\!\!\twoheadrightarrow A$ *then there exists some call-by-value answer* A_0 *such that* $M \mapsto_{\mathsf{Val}}\!\!\!\!\twoheadrightarrow A_0$.

Proposition 1.3 *Properties of call-by-need [2, 3, 18, 19]:*
- Need *satisfies subject reduction.*
- Need *is confluent.*
- *If* $M \xrightarrow[\mathsf{Need}]{}\!\!\!\!\twoheadrightarrow A$ *then there exists some call-by-need answer* A_0 *such that* $M \mapsto_{\mathsf{Need}}\!\!\!\!\twoheadrightarrow A_0$.

1.2 Unitary linear calculi

Figure 3 presents the (unitary) linear lambda calculus Lin. Types here are base types, linear function or modal types; members of the latter are expected to reduce to an expression prefixed with the ! operator. Terms, aside from the usual variables, abstractions and applications, can be prefixed with the ! or be an eliminator to access the body of such a prefixed subterm. This system is discussed is detail by Wadler [25] and in our previous work [20]; we summarize its properties presently.

Proposition 1.4 *Properties of linear lambda [18, 20]:*
- Lin *satisfies subject reduction.*
- Lin *is confluent.*
- *If* $M \xrightarrow[\mathsf{Lin}]{}\!\!\!\!\twoheadrightarrow A$ *then there exists some linear lambda answer* A_0 *such that* $M \mapsto_{\mathsf{Lin}}\!\!\!\!\twoheadrightarrow A_0$.

Figures 4 and 5 present the standard and steadfast Girard translations for call-by-name and call-by-value, respectively.

Proposition 1.5 *The translation* $\circ$ *from* Name *to* Lin *preserves substitution, typing judgments, reduction and evaluation [18, 20]:*

Syntactic domains As for Name and Val, plus:

Terms $L, M, N ::= \cdots \mid \text{let } x = M \text{ in } N$

Typing judgments As for Name and Val, plus:

$$\frac{, \vdash M : A \qquad \Delta, x : A \vdash N : B}{, , \Delta \vdash \text{let } x = M \text{ in } N : B}\ \text{Let}$$

Call-by-need reduction

(I)	$(\lambda x.\, M)\ N$	$\xrightarrow[\text{Need}]{}$	let $x = N$ in M
(V)	let $x = V$ in M	$\xrightarrow[\text{Need}]{}$	$M[x := V]$
(C)	(let $x = L$ in M) N	$\xrightarrow[\text{Need}]{}$	let $x = L$ in $(M\ N)$
(A)	let $y =$ (let $x = L$ in M) in N	$\xrightarrow[\text{Need}]{}$	let $x = L$ in (let $y = M$ in N)
(G)	let $x = M$ in N	$\xrightarrow[\text{Need}]{}$	N, if x is not free in N

Call-by-need evaluation

Evaluation contexts $E ::= [\,] \mid E\ M \mid \text{let } x = M \text{ in } E \mid \text{let } x = E \text{ in } E[x]$
Answers $A ::= \lambda x.\, M \mid \text{let } x = M \text{ in } A$

(I)	$(\lambda x.\, M)\ N$	$\xrightarrow[\text{Need}]{}$	let $x = N$ in M
(V)	let $x = (\lambda y.\, M)$ in $E[x]$	$\xrightarrow[\text{Need}]{}$	$(E[x])[x := \lambda y.\, M]$
(C)	(let $x = M$ in A) N	$\xrightarrow[\text{Need}]{}$	let $x = M$ in $(A\ N)$
(A)	let $y =$ (let $x = M$ in A) in $E[x]$	$\xrightarrow[\text{Need}]{}$	let $x = M$ in (let $y = A$ in $E[x]$)

Figure 2: The call-by-need lambda calculus.

Standard translation

Of types

$$\begin{aligned} Z^\circ &\equiv Z \\ (A \to B)^\circ &\equiv (!A^\circ) \multimap B^\circ \end{aligned}$$

Of terms

$$\begin{aligned} x^\circ &\equiv x \\ (\lambda x.\, M)^\circ &\equiv \lambda y.\, \text{let } !x = y \text{ in } M^\circ \\ (M\ N)^\circ &\equiv M^\circ\ !N^\circ \end{aligned}$$

Of typing contexts

$$(, , x : A)^\circ \equiv , ^\circ, x : A^\circ$$

Figure 4: The unitary standard translation.

(i) $(M[x := N])^\circ \equiv M^\circ[x := N^\circ]$.

(ii) $, \vdash_{\mathsf{Name}} M : A$ *if and only if* $, ^\circ; - \vdash_{\mathsf{Lin}} M^\circ : A^\circ$.

(iii) $M \xrightarrow[\mathsf{Name}]{}\!\!\!\!\rightarrow N$ *if and only if* $M^\circ \xrightarrow[\mathsf{Lin}]{}\!\!\!\!\rightarrow N^\circ$.

(iv) $M \mapsto\!\!\!\!\xrightarrow[\mathsf{Name}]{}\!\!\!\!\rightarrow N$ *if and only if* $M^\circ \mapsto\!\!\!\!\xrightarrow[\mathsf{Lin}]{}\!\!\!\!\rightarrow N^\circ$.

Proposition 1.6 *The translation* $*$ *from* Val *to* Lin *preserves substitution of values, typing judgments, reduction and evaluation [18, 20].*

(i) $(M[x := V])^* \equiv M^*[x := V^+]$.

(ii) $, \vdash_{\mathsf{Val}} M : A$ *if and only if* $, ^*; - \vdash_{\mathsf{Lin}} M^* : A^*$, *and* $, \vdash_{\mathsf{Val}} V : A$ *if and only if* $, ^*; - \vdash_{\mathsf{Lin}} V^+ : A^+$.

(iii) If $M \xrightarrow[\mathsf{Val}]{}\!\!\!\!\rightarrow N$ *then* $M^* \xrightarrow[\mathsf{Lin}]{}\!\!\!\!\rightarrow N^*$.

(iv) $M \mapsto\!\!\!\!\xrightarrow[\mathsf{Val}]{}\!\!\!\!\rightarrow N$ *if and only if* $M^* \mapsto\!\!\!\!\xrightarrow[\mathsf{Lin}]{}\!\!\!\!\rightarrow N^*$.

Steadfast translation

Of types

$$\begin{aligned} A^* &\equiv !A^+ \\ Z^+ &\equiv Z \\ (A \to B)^+ &\equiv (A^* \multimap B^*) \end{aligned}$$

Of terms

$$\begin{aligned} V^* &\equiv !V^+ \\ (M\ N)^* &\equiv (\text{let } !z = M^* \text{ in } z)\ N^* \\ (\text{let } x = M \text{ in } N)^* &\equiv \text{let } !x = M^* \text{ in } N^* \\ x^+ &\equiv x \\ (\lambda x.\, M)^+ &\equiv \lambda y.\, \text{let } !x = y \text{ in } M^* \end{aligned}$$

Of typing contexts

$$(, , x : A)^* \equiv , ^*, x : A^+$$

Figure 5: The unitary steadfast translation.

As a counterexample to completeness in Proposition 1.6.(iii) we can take $M \equiv I\ P\ M$ and $N \equiv P\ M$ where I is the identity function $I \equiv \lambda x.\, x$ and P is any non-value. Then clearly $M \not\to N$ in Val, but we do have in Lin:

$$\begin{aligned} & (I\ P\ M)^* \\ \equiv\ & (\text{let } !w = (\text{let } !z = !I^+ \text{ in } z)\ P^* \text{ in } w)\ M^* \\ \xrightarrow[(\beta!)]{}\ & (\text{let } !w = (\lambda y.\, \text{let } !x = y \text{ in } !x)\ P^* \text{ in } w)\ M^* \\ \xrightarrow[(\beta \multimap)]{}\ & (\text{let } !w = (\text{let } !x = P^* \text{ in } !x) \text{ in } w)\ M^* \\ \xrightarrow[(!!)]{}\ & (\text{let } !x = P^* \text{ in let } !w = !x \text{ in } w)\ M^* \\ \xrightarrow[(\beta!)]{}\ & (\text{let } !x = P^* \text{ in } x)\ M^* \end{aligned}$$

Syntactic domains

Types	A, B	$::=$	$Z \mid !A \mid A \multimap B$
Terms	L, M, N	$::=$	$x \mid !M \mid \text{let } !x = M \text{ in } N \mid \lambda x.\, M \mid M\ N$

Typing judgments

The typing axiom

$$\frac{}{-;\ x : A \vdash x : A}\ \text{Id}$$

Substructure rules

$$\frac{\Phi,\ y : A,\ z : A;\ ,\ \vdash M : B}{\Phi,\ x : A;\ ,\ \vdash M[y := x, z := x] : B}\ \text{Contraction}$$

$$\frac{\Phi;\ ,\ \vdash M : B}{\Phi,\ x : A;\ ,\ \vdash M : B}\ \text{Weakening} \qquad \frac{\Phi;\ x : A,\ ,\ \vdash M : B}{\Phi,\ x : A;\ ,\ \vdash M : B}\ \text{Dereliction}$$

Term structure rules

$$\frac{\Phi;\ -\vdash M : A}{\Phi;\ -\vdash\ !M :!A}\ !\text{-I} \qquad \frac{\Phi;\ ,\ \vdash M :!A \qquad \Psi,\ x : A;\ \Delta \vdash N : B}{\Phi,\ \Psi;\ ,\ ,\ \Delta \vdash \text{let } !x = M \text{ in } N : B}\ !\text{-E}$$

$$\frac{\Phi;\ ,\ ,\ x : A \vdash M : B}{\Phi;\ ,\ \vdash \lambda x.\, M : A \multimap B}\ \multimap\text{-I} \qquad \frac{\Phi;\ ,\ \vdash M : A \multimap B \qquad \Psi;\ \Delta \vdash N : A}{\Phi,\ \Psi;\ ,\ ,\ \Delta \vdash M\ N : B}\ \multimap\text{-E}$$

Linear lambda reduction

$(\beta\multimap)$	$(\lambda x.\, M)\ N$	$\xrightarrow[\text{Lin}]{}$	$M[x := N]$
$(\beta!)$	$\text{let } !x = !M \text{ in } N$	$\xrightarrow[\text{Lin}]{}$	$N[x := M]$
$(!\multimap)$	$(\text{let } !x = L \text{ in } M)\ N$	$\xrightarrow[\text{Lin}]{}$	$\text{let } !x = L \text{ in } (M\ N)$
$(!!)$	$\text{let } !y = (\text{let } !x = L \text{ in } M) \text{ in } N$	$\xrightarrow[\text{Lin}]{}$	$\text{let } !x = L \text{ in } (\text{let } !y = M \text{ in } N)$

Linear lambda evaluation

Evaluation contexts	E	$::=$	$[\,] \mid E\ M \mid \text{let } !x = E \text{ in } M$	
Answers	A	$::=$	$x \mid \lambda x.\, M \mid !M$	

$(\beta\multimap)$	$(\lambda x.\, M)\ N$	$\xrightarrow[\text{Lin}]{}$	$M[x := N]$
$(\beta!)$	$\text{let } !x = !N \text{ in } M$	$\xrightarrow[\text{Lin}]{}$	$M[x := N]$

Figure 3: The linear lambda calculus.

$\equiv \quad (P\ M)^*$

So clearly completeness does not hold there. Completeness does hold for standard reduction, where the $(!{-\circ}, !!)$ rules do not appear.

Figure 6 presents the affine lambda calculus Aff. This system is just like Lin except that we allow weakening anywhere, and regulate only contraction with the modal operator. This change manifests itself in two places. First, in the Weakening typing rule, the new variable is added to the linear, rather than the intuitionistic zone. Thus in conjunction with Dereliction a weakened variable can be used in any manner whatsoever in typing a term. Secondly, we have a new reduction rule (!W), which allows any subterm bound to a variable introduced by Weakening — that is, one which does not appear free in the body of an eliminator — to be simply discarded without further evaluation, even without an exposed prefix.

Proposition 1.7 *Properties of affine lambda [18, 20]:*
- Aff *satisfies subject reduction.*
- Aff *is confluent.*
- *If* $M \xrightarrow[\mathsf{Aff}]{}\!\!\!\!\rightarrow\!\!\!\!\rightarrow A$ *then there exists some affine lambda answer* A_0 *such that* $M \vdash_{\mathsf{Aff}} A_0$.

The shift from linear to affine lambda as the target of the steadfast translation corresponds to a shift from a strict to a lazy evaluation semantics in the source calculus. In other words, we map not from call-by-value into affine lambda, but rather from call-by-need.

Proposition 1.8 *The translation* $*$ *from* Need *to* Aff *preserves substitution of values, typing judgments, reduction and evaluation [18, 20].*

(i) $(M[x := V])^* \equiv M^*[x := V^+]$.

(ii) $, \vdash_{\mathsf{Need}} M : A$ *if and only if* $, ^*; - \vdash_{\mathsf{Aff}} M^* : A^*$, *and* $, \vdash_{\mathsf{Need}} V : A$ *if and only if* $, ^* \vdash_{\mathsf{Aff}} V^+ : A^+$.

(iii) If $M \xrightarrow[\mathsf{Need}]{}\!\!\!\!\rightarrow\!\!\!\!\rightarrow N$ *then* $M^* \xrightarrow[\mathsf{Aff}]{}\!\!\!\!\rightarrow\!\!\!\!\rightarrow N^*$.

(iv) $M \vdash_{\mathsf{Need}} N$ *if and only if* $M^* \vdash_{\mathsf{Aff}} N^*$.

Unfortunately, we can make no simple statement about the translation of call-by-need into Lin instead of Aff, since we could not expect to soundly translate both (V) and (G) steps. This discrepancy motivates the finer control obtained by separating weakening from contraction, which we explore below.

2 The separated-linear lambda calculus

Figure 7 presents the types, terms and main reduction rules of the separated linear lambda calculus SLin. A type is a base type, a linear function, or one of two modal types. As in Name and Val, basic terms include variables, function abstractions and applications. We have a number of new term forms, however, for the introduction and elimination of the new modal operators. The prefixes $!^w$ and $!^c$ denote terms which one may discard or duplicate, respectively; we refer to each prefix as the *dual* of the other. The corresponding $!^w$- and $!^c$-eliminators strip prefixes from expressions for discarding, or for copying into another term. However, the remaining eliminator has no corresponding prefix, and as such is a bit mysterious until one considers the structure of the typing rules.

Typing judgments in our previous work on Lin divided the typing environment into two zones: one zone held *linear* assumptions, on the variables of which no weakening or contraction was allowed, while the other held *intuitionistic* assumptions, on whose variables the weakening and contraction rules could be applied. The elegance of such a duality of assumptions has been noted for full linear logic in general by Girard [13], and for the intuitionistic fragment in particular by Barber [4]. Here, since we want to separately enable weakening and contraction, we will have a total of four zones: in addition to the linear and intuitionistic zones, we will have one zone which admits weakening but not contraction, and one zone which admits contraction but not weakening.

We let $\mathcal{E}$ and $\mathcal{F}$ range over such quadruples $, ; \Delta; \Phi; \Psi$ of typing assumptions. We find it rather confusing to write judgments with assumptions explicitly separated, and instead we adopt the notation proposed by Jacobs [16]: we write $\mathcal{E}, ^{\blacktriangleright_i} x : A$ to mean the quadruple $\mathcal{E}$ extended by adding the assumption $x : A$ to its collection of linear assumptions, and similarly $^{\blacktriangleright_w}$, $^{\blacktriangleright_c}$ and $^{\blacktriangleright_i}$ to extend respectively the zones which admit weakening, contraction and both.

The role of the "extra" eliminator structure is now clear: we must be able to eliminate variables from each of the four zones, and this eliminator corresponds to the intuitionistic zone. We have no intuitionistic prefix because such an amalgamation of the $!^w$ and $!^c$ prefixes is exactly what we are trying to avoid with this calculus. We wish to enable weakening and contraction separately from each other, and so an intuitionistic term is one marked as both weakenable and contractable: that is, of either form $!^w!^c M$ or $!^c!^w M$; the $!^i$-eliminator does not distinguish between the two.

The typing rules are straightforward. The Id rule introduces a linear variable. We do not have introduction rules for other zones, but rather, the Dereliction rules. Since maintaining an assumption in (for example) the linear zone entails *not* performing certain activities, we can correctly move an assumption from a more restrictive zone to a less restrictive zone at any time, but not vice-versa. While the variable in the subject of the derelicted assumption will not (yet) be used in the newly enabled manner, the typing is still appropriate, and we can subsequently apply the newly enabled rules. We have two each of the Weakening and Contraction rules: one where the operation is performed from a zone where the other operation is not allowed (for example, the Weakening rule adds a variable into the zone where weakening but not contraction is allowed) and one where the operation is performed in the intuitionistic zone. Introduction of a modal prefix is allowed only if a term contains no free variables on which the

Syntactic domains As for Lin.

Typing judgments As for Lin, but changing Weakening as follows:

$$\frac{\Phi;\ ,\ \vdash M : B}{\Phi;\ ,\ ,\ x : A \vdash M : B}\ \text{Weakening}$$

Affine lambda reduction As for Lin, plus the following:

$$(!\mathrm{W}) \qquad \text{let } !x = M \text{ in } N \ \to\ N, \qquad \text{if } x \text{ is not free in } N$$

Affine lambda evaluation

Evaluation contexts	E	$::=$	$[\,] \mid E\,M \mid !E \mid \text{let } !x = M \text{ in } E \mid \text{let } !x = E \text{ in } E[x]$
Answers	A	$::=$	$x \mid \lambda x.\,M \mid !A \mid \text{let } !x = M \text{ in } A$

$(\beta\multimap)$	$(\lambda x.\,M)\,N$	$\xrightarrow[\mathsf{Aff}]{}$	$M[x := N]$
$(\beta!)$	$\text{let } !x = !M \text{ in } E[x]$	$\xrightarrow[\mathsf{Aff}]{}$	$(E[x])[x := M]$
$(!\multimap)$	$(\text{let } !x = M \text{ in } A)\,N$	$\xrightarrow[\mathsf{Aff}]{}$	$\text{let } !x = M \text{ in } (A\,N)$
$(!!)$	$\text{let } !y = (\text{let } !x = M \text{ in } A) \text{ in } E[y]$	$\xrightarrow[\mathsf{Aff}]{}$	$\text{let } !x = M \text{ in } (\text{let } !y = A \text{ in } E[y])$

Figure 6: The affine lambda calculus.

operation enabled by the prefix is not allowed. For example, the $!^w$-I rule requires that the contractable and linear zones be empty; a term under a $!^w$ prefix should be discardable, but if such a free variable were discarded we would violate the typing assumptions.

The reduction rules for SLin are given in Figure 8. The main reduction rules for SLin are the five β rules and the weakening rule. As usual, we have a β rule whenever an elimination rule occurs immediately below the corresponding introduction rule; we have two such rules for the $!^i$-eliminator since the two prefixes to be eliminated may arrive in either order. The weakening rule allows us to take advantage of the fact that a term prefixed by $!^w$ may be discarded immediately if it is not to be used, without necessarily finding a $!^c$ prefix for a $!^i$-eliminator.

The remaining reduction rules are commuting rules for managing the additional structure. The rôle of the $(!\multimap)$ and $(!!)$ rules are the same as for the rules of the same name in Lin. In the left-hand sides of the former, a modal elimination rule can block a $(\beta\multimap)$ step by hiding an abstraction in the left subterm; in the left-hand sides of the latter, a modal elimination rule can block a $(\beta!)$ step by hiding a correctly prefixed term. In both cases, the rule expands the scope of the variable bound by the blocking eliminator to allow any blocked reduction at that point. The $(!!!)$ rules extend the $(!!)$ rules to the cases where an amalgamated eliminator "sees" one of the two modal prefixes required for a reduction step, but under that prefix another binding can hide the second prefix. Again, the $(!!!)$ rules simply expand the scope of the inner binding and remove it from a position where it may no longer block the reduction of the outer binding.

Some rules might seem to be missing from the $(!!!)$ set, for example a rule:

$$\begin{aligned}&\text{let } !^i x = !^w(\text{let } !^c y = !^c L \text{ in } M) \text{ in } N\\ &\quad\to\ \text{let } !^c y = !^c L \text{ in let } !^i x = !^w M \text{ in } N\ . \qquad (1)\end{aligned}$$

This rule — as would the similar rule with the $!^c$'s and $!^w$'s exchanged — disallows the subject reduction property. The reason is as follows: First, it may be that x does not appear free in N, and so for the counterexample we assume so; we moreover take $M \equiv !^c M_0$. So on the left-hand side we have some typing as shown in Figure 9. However, L is not discardable: possibly, L will reduce to an expression $!^w L_0$ whose prefix can be removed and the body discarded, but nonetheless L by itself cannot in general be discarded. So when trying to type the right-hand side of (1), we fail because the variable y which eliminates the prefix from L prevents the addition of the $!^w$ prefix to $!^c M_0$. The problem becomes clearer if we allow the $(\beta!^i)$ contraction of the new inner eliminator: since $x \notin \mathrm{fv}(N)$ we have

$$\begin{aligned}&\text{let } !^c y = !^c L \text{ in } (\text{let } !^i x = !^w !^c M_0 \text{ in } N)\\ \xrightarrow[(\beta!^i)]{}\ &\text{let } !^c y = !^c L \text{ in } N\ ,\end{aligned}$$

and since $y \notin \mathrm{fv}(N)$ as well, we would clearly have to perform weakening on a variable which is exclusively contractable. These problems do not arise for the $(!!)$ rules, since (for example) they would not move a $!^c$-eliminator from under a $!^w$ prefix.

Failure of subject reduction does not explain all of the absent possible $(!!!)$ rules. The step

$$\begin{aligned}&\text{let } !^i x = !^w(\text{let } !^w y = !^w L \text{ in } M) \text{ in } N\\ &\quad\to\ \text{let } !^w y = !^w L \text{ in let } !^i x = !^w M \text{ in } N\end{aligned}$$

as well as the similar rule with $!^c$ prefixes rather than the $!^w$'s both preserve subject reduction. We omit these rules because we see no use for them. They are not essential for any of the properties we seek, neither of the calculus itself nor of any mapping into SLin from another system. This uselessness is not entirely surprising, since the inner subterm $(\text{let } !^w y = !^w L \text{ in } M)$ is already a (much more interesting) $\beta!^i$ redex; since it can be contracted immediately, it is hard to view as in

Syntactic domains

$$
\begin{array}{lrll}
\text{Types} & A, B & ::= & Z \mid A \multimap B \mid !^w A \mid !^c A \\
\text{Terms} & L, M, N & ::= & x \mid \lambda x.\, M \mid M\ N \\
 & & \mid & \text{let } !^w x = M \text{ in } N \mid !^w M \\
 & & \mid & \text{let } !^c x = M \text{ in } N \mid !^c M \\
 & & \mid & \text{let } !^i x = M \text{ in } N
\end{array}
$$

Typing rules

The typing axiom

$$\frac{}{{}^{\blacktriangleright_l} x : A \vdash x : A}\ \text{Id}$$

Substructure rules

$$\frac{\mathcal{E}, {}^{\blacktriangleright_c} x : A, y : A \vdash M : B}{\mathcal{E}, {}^{\blacktriangleright_c} z : A \vdash M[x := z, y := z] : B}\ \text{Contraction} \qquad \frac{\mathcal{E} \vdash M : B}{\mathcal{E}, {}^{\blacktriangleright_w} x : A \vdash M : B}\ \text{Weakening}$$

$$\frac{\mathcal{E}, {}^{\blacktriangleright_i} x : A, y : A \vdash M : B}{\mathcal{E}, {}^{\blacktriangleright_i} z : A \vdash M[x := z, y := z] : B}\ \text{Contraction/W} \qquad \frac{\mathcal{E} \vdash M : B}{\mathcal{E}, {}^{\blacktriangleright_i} x : A \vdash M : B}\ \text{Weakening/C}$$

$$\frac{\mathcal{E}, {}^{\blacktriangleright_l} x : A \vdash M : B}{\mathcal{E}, {}^{\blacktriangleright_w} x : A \vdash M : B}\ \text{Dereliction}!^w \qquad \frac{\mathcal{E}, {}^{\blacktriangleright_l} x : A \vdash M : B}{\mathcal{E}, {}^{\blacktriangleright_c} x : A \vdash M : B}\ \text{Dereliction}!^c$$

$$\frac{\mathcal{E}, {}^{\blacktriangleright_c} x : A \vdash M : B}{\mathcal{E}, {}^{\blacktriangleright_i} x : A \vdash M : B}\ \text{Dereliction}!^w/C \qquad \frac{\mathcal{E}, {}^{\blacktriangleright_w} x : A \vdash M : B}{\mathcal{E}, {}^{\blacktriangleright_i} x : A \vdash M : B}\ \text{Dereliction}!^c/W$$

Term structure rules

$$\frac{{}^{\blacktriangleright_i}, , {}^{\blacktriangleright_c}\Phi \vdash M : A}{{}^{\blacktriangleright_i}, , {}^{\blacktriangleright_c}\Phi \vdash M : !^c A}\ !^c\text{-I} \qquad \frac{\mathcal{E} \vdash M : !^c A \qquad \mathcal{F}, {}^{\blacktriangleright_c} x : A \vdash N : B}{\mathcal{E}, \mathcal{F} \vdash \text{let } !^c x = M \text{ in } N : B}\ !^c\text{-E}$$

$$\frac{{}^{\blacktriangleright_i}, , {}^{\blacktriangleright_w}\Phi \vdash M : A}{{}^{\blacktriangleright_i}, , {}^{\blacktriangleright_w}\Phi \vdash M : !^w A}\ !^w\text{-I} \qquad \frac{\mathcal{E} \vdash M : !^w A \qquad \mathcal{F}, {}^{\blacktriangleright_w} x : A \vdash N : B}{\mathcal{E}, \mathcal{F} \vdash \text{let } !^w x = M \text{ in } N : B}\ !^w\text{-E}$$

$$\frac{\mathcal{E} \vdash M : !^w !^c A \qquad \mathcal{F}, {}^{\blacktriangleright_i} x : A \vdash N : B}{\mathcal{E}, \mathcal{F} \vdash \text{let } !^i x = M \text{ in } N : B}\ !^i\text{-E}^a \qquad \frac{\mathcal{E} \vdash M : !^c !^w A \qquad \mathcal{F}, {}^{\blacktriangleright_i} x : A \vdash N : B}{\mathcal{E}, \mathcal{F} \vdash \text{let } !^i x = M \text{ in } N : B}\ !^i\text{-E}^b$$

$$\frac{\mathcal{E}, {}^{\blacktriangleright_l} x : A \vdash M : B}{\mathcal{E} \vdash \lambda x.\, M : A \multimap B}\ \multimap\text{-I} \qquad \frac{\mathcal{E} \vdash M : A \multimap B \qquad \mathcal{F} \vdash N : A}{\mathcal{E}, \mathcal{F} \vdash M\ N : B}\ \multimap\text{-E}$$

Figure 7: Syntax and typing rules for the separated linear lambda calculus.

β-reduction rules

$$
\begin{array}{lll}
(\beta\multimap) & (\lambda x.\, M)\ N & \xrightarrow[\mathsf{SLin}]{} \quad M[x := N] \\
(\beta!^{c}) & \text{let } !^{c}x = !^{c}M \text{ in } N & \xrightarrow[\mathsf{SLin}]{} \quad N[x := M] \\
(\beta!^{w}) & \text{let } !^{w}x = !^{w}M \text{ in } N & \xrightarrow[\mathsf{SLin}]{} \quad N[x := M] \\
(\beta!^{i}_{cw}) & \text{let } !^{i}x = !^{w}!^{c}M \text{ in } N & \xrightarrow[\mathsf{SLin}]{} \quad N[x := M] \\
(\beta!^{i}_{wc}) & \text{let } !^{i}x = !^{c}!^{w}M \text{ in } N & \xrightarrow[\mathsf{SLin}]{} \quad N[x := M]
\end{array}
$$

Weakening rule

$$
(\mathrm{W}!^{i}) \qquad \text{let } !^{i}x = !^{w}M \text{ in } N \quad \xrightarrow[\mathsf{SLin}]{} \quad N,\ x \notin \mathrm{fv}(N)
$$

Commuting conversion rules

$$
\begin{array}{lll}
(!^{\square}\multimap) & (\text{let } !^{\square}x = L \text{ in } M)\ N \quad \text{where } !^{\square} \text{ ranges over } !^{w},\ !^{c},\ !^{i}. & \xrightarrow[\mathsf{SLin}]{} \quad \text{let } !^{\square}x = L \text{ in } (M\ N) \\
(!^{\square}!^{\triangle}) & \text{let } !^{\square}x = (\text{let } !^{\triangle}y = L \text{ in } M) \text{ in } N \quad \text{where } !^{\square} \text{ and } !^{\triangle} \text{ each range over } !^{w},\ !^{c},\ !^{i}. & \xrightarrow[\mathsf{SLin}]{} \quad \text{let } !^{\triangle}y = L \text{ in (let } !^{\square}x = M \text{ in } N) \\
(!^{i}!^{\square}!^{w}) & \text{let } !^{i}x = !^{\square}(\text{let } !^{i}y = !^{w}L \text{ in } M) \text{ in } N \quad \text{where } !^{\square} \text{ ranges over } !^{w} \text{ and } !^{c}. & \xrightarrow[\mathsf{SLin}]{} \quad \text{let } !^{i}y = !^{w}L \text{ in let } !^{i}x = !^{\square}M \text{ in } N
\end{array}
$$

Figure 8: Reduction in the separated linear lambda calculus.

Well-typed left-hand side

$$
\dfrac{\dfrac{\dfrac{\vdots\ \mathcal{P}_1}{\blacktriangleright_i,\ \vdash L : A_0}}{\blacktriangleright_i,\ \vdash !^{c}L : !^{c}A_0}\,!^{c}\text{-I} \qquad \dfrac{\vdots\ \mathcal{P}_2}{\blacktriangleright_i \Delta, \blacktriangleright_c y : A_0 \vdash M_0 : B}\,!^{c}\text{-I}}{\blacktriangleright_i \Delta, \blacktriangleright_c y : A_0 \vdash !^{c}M_0 :!^{c}B}\,!^{c}\text{-E}}{\blacktriangleright_i \Delta \vdash \text{let } !^{c}y = !^{c}L \text{ in } !^{c}M_0 :!^{c}B}\,!^{w}\text{-I} \qquad \dfrac{\vdots\ \mathcal{P}_3}{\mathcal{E} \vdash N : A}\,\text{Weakening/C}}{\mathcal{E}, \blacktriangleright_i, , \Delta \vdash \text{let } !^{i}x = !^{w}(\text{let } !^{c}y = !^{c}L \text{ in } !^{c}M_0) \text{ in } N : A}\,!^{i}\text{-E}_1
$$

Ill-typed right-hand side

$$
\dfrac{\dfrac{\vdots\ \mathcal{P}_1}{\blacktriangleright_i,\ \vdash L : A_0}\,!^{c}\text{-I}}{\blacktriangleright_i,\ \vdash !^{c}L : !^{c}A_0} \qquad \dfrac{\dfrac{\dfrac{\vdots\ \mathcal{P}_2}{\blacktriangleright_i \Delta, \blacktriangleright_c y : A_0 \vdash M_0 : B}\,!^{c}\text{-I}}{\blacktriangleright_i \Delta, \blacktriangleright_c y : A_0 \vdash !^{c}M_0 : !^{c}B}\,(\text{Illegal step})}{\blacktriangleright_i \Delta, \blacktriangleright_c y : A_0 \vdash !^{w}!^{c}M_0 : !^{w}!^{c}B} \qquad \dfrac{\vdots\ \mathcal{P}_3}{\mathcal{E} \vdash N : A}\,\text{Weakening/C}}{\mathcal{E}, \blacktriangleright_i \Delta, \blacktriangleright_c y : A_0 \vdash \text{let } !^{i}x = !^{w}!^{c}M_0 \text{ in } N : A}\,!^{i}\text{-E}_1
$$

$$
\mathcal{E}, \blacktriangleright_i, , \Delta \vdash \text{let } !^{c}y = !^{c}L \text{ in let } !^{i}x = !^{w}!^{c}M_0 \text{ in } N : A
$$

Figure 9: Example of the failure of subject reduction in an omitted (!!!) rule.

the way! The rule

$$\begin{array}{l}\text{let } !^i x = !^w(\text{let } !^i y = !^c L \text{ in } M) \text{ in } N \\ \quad \rightarrow \quad \text{let } !^i y = !^c L \text{ in let } !^i x = !^w M \text{ in } N\end{array}$$

also turns out not to be required, but more importantly it could complicate standard reduction, by changing the evaluation status of a term. We return to this point in Appendix B, in the context of the point in the proof of standardization where it would cause a problem, following B.10.

One further suspicion can be raised about the (!!!) rules: why do we require the innermost prefix on the subterms L? Although their presence is somewhat unattractive, without them we lose confluence. Consider the less restrictive rule

$$\begin{array}{l}\text{let } !^i x = !^w(\text{let } !^i y = L \text{ in } M) \text{ in } N \\ \quad \rightarrow \quad \text{let } !^i y = L \text{ in let } !^i x = !^w M \text{ in } N\end{array}$$

and the term

$$\text{let } !^i x = !^w(\text{let } !^i y = (\text{let } !^c z = L_0 \text{ in } L_1) \text{ in } M) \text{ in } N \ .$$

We can apply the new rule at the top level, and as the $(!^i!^c)$ rule to the subterm

$$\text{let } !^i y = (\text{let } !^c z = L_0 \text{ in } L_1) \text{ in } M \ .$$

After the latter contraction we can no longer apply a (!!!) rule at the top level, since we have just restricted

$$\text{let } !^i x = !^w(\text{let } !^c z = L_0 \text{ in let } !^i y = L_1 \text{ in } M) \text{ in } N$$

from being the left-hand side of a (!!!) rule for the sake of subject reduction. Since we could have for example $L_0 \equiv w$ for some free variable w, confluence would not hold in general.

Perhaps the biggest weakness of SLin is its complexity. Many presentations of λ avoid any mention of the substructure typing rules at all, and certainly we require more rules for juggling the new terms structures than what one might ideally prefer. Still, we believe that the calculus as we present it here represents an acceptable compromise between simplicity and expressiveness. If one wishes to control substructural operations, one must have some mechanism for doing so, and furthermore separate mechanisms for separate control. We have relegated most of this mechanism to the typing rules; while the commuting conversion reductions are somewhat numerous, they are less so than in most other (not even separated) linear calculi.

These difficulties aside, we can now assert the basic properties of general reduction in SLin.

2.1 Substitution properties

Lemma 2.1 *Typing proofs in* SLin *are well-behaved under substitution.*

1. *Let* $\mathcal{E}, {}^{\blacktriangleright_\iota}x : B \vdash M : A$ *and* $\mathcal{F} \vdash N : B$. *Then* $\mathcal{E}, \mathcal{F} \vdash M[x := N] : A$.
2. *Let* $\mathcal{E}, {}^{\blacktriangleright_c}x : B \vdash M : A$ *and* ${}^{\blacktriangleright_i}, , {}^{\blacktriangleright_c}\Delta \vdash N : B$. *Then* $\mathcal{E}, {}^{\blacktriangleright_i}, , {}^{\blacktriangleright_c}\Delta \vdash M[x := N] : A$.
3. *Let* $\mathcal{E}, {}^{\blacktriangleright_w}x : B \vdash M : A$ *and* ${}^{\blacktriangleright_i}, , {}^{\blacktriangleright_w}\Delta \vdash N : B$. *Then* $\mathcal{E}, {}^{\blacktriangleright_i}, , {}^{\blacktriangleright_w}\Delta \vdash M[x := N] : A$.
4. *Let* $\mathcal{E}, {}^{\blacktriangleright_i}x : B \vdash M : A$ *and* ${}^{\blacktriangleright_i}, \vdash N : B$. *Then* $\mathcal{E}, {}^{\blacktriangleright_i}, \vdash M[x := N] : A$.

Lemma 2.2 *Let* $L, M, N \in$ SLin *where* $M \xrightarrow[\text{SLin}]{} N$. *Then*

1. $L[x := M] \xrightarrow[\text{SLin}]{}\!\!\!\!\!\rightarrow L[x := N]$ *and*
2. $M[x := L] \xrightarrow[\text{SLin}]{} N[x := L]$.

2.2 Subject reduction

With the above properties of substitution, subject reduction follows easily.

Proposition 2.3 SLin *satisfies the subject reduction property.*

Proof: By induction on the depth of the typing tree. For the β rules we use Lemma 1; for the other rules we need simply rearrange the typing trees. ♦

2.3 Confluence

Confluence follows from the technical developments of Appendix A, and we remove the proof to the end of that section.

Proposition 2.4 SLin *is confluent.*

2.4 Standard evaluation

The notion of evaluation in SLin fulfills a prediction made by Jacobs [16]: that although the order of application of $!^c$ and $!^w$ may be indistinguishable in many sensible models, the terms $!^c!^w M$ and $!^w!^c M$ may reasonably have different operational interpretations[1]. The difference is in the evaluation of the $!^i$-eliminator, whose evaluation depends on the order in which the two prefixes are encountered. When the $!^c$ prefix is outermost, we reduce its subterm to find the expected $!^w$ prefix, and immediately substitute its body for the variable bound by the eliminator: thus the evaluation context forms

$$E ::= \cdots \mid \text{let } !^i x = E \text{ in } M \mid \text{let } !^i x = !^c E \text{ in } M \ .$$

Conversely, when the $!^w$ prefix is outermost, we delay any further evaluation of the subterm under the prefix, and instead proceed with the evaluation of the body of

[1] To be specific: Jacobs wrote that a term of the form $!^c!^w M$ would be "less efficient, because it involves doing nothing many times" than one of the form $!^w!^c M$, although here the operational interpretation of neither ordering of the prefixes seems necessarily more efficient than the other. In his development of the models, he first presents separate models where one operator distributes over the other but not vice-versa, and then describes a model where the two combinations are essentially equivalent.

Separated standard translation

Of types

$$\begin{aligned}(A \to B)^\circ &\equiv (!^c!^w A^\circ) \multimap B^\circ \\ Z^\circ &\equiv Z\end{aligned}$$

Of terms

$$\begin{aligned}x^\circ &\equiv x \\ (\lambda x.\, M)^\circ &\equiv \lambda y.\, \text{let } !^i x = y \text{ in } M^\circ \\ (M\ N)^\circ &\equiv M^\circ\ (!^c!^w N^\circ)\end{aligned}$$

Of typing environments

$$(,\, , x : A)^\circ \equiv\ ,^\circ, x : A^\circ$$

Figure 11: The separated Girard translations into SLin.

the eliminator. Only when the eliminator-bound variable is demanded do we resume evaluation under the $!^w$ prefix: this strategy is manifested in the evaluation contexts

$$\begin{aligned}E ::= \cdots \mid\ &\text{let } !^i x = E \text{ in } M \\ \mid\ &\text{let } !^i x = !^w M \text{ in } E \\ \mid\ &\text{let } !^i x = !^w E \text{ in } E[x]\ \ ,\end{aligned}$$

which are reminiscent of Need evaluation.

Arguably, we ought to treat the $!^w$-eliminator is the same manner: disallowing the $(\beta!^w)$ rule from evaluation, and including an evaluation context of the form

$$\text{let } !^i x = !^w M \text{ in } E\ \ .$$

Ultimately there is little difference between the two. Since such an x appears at most once in the body of the eliminator, we are not duplicating work when we apply $(\beta!^w)$: in either case, M would be evaluated only when x is demanded. Moreover, it will become clear that neither configuration would affect the translations of the intuitionistic systems into SLin.

Proposition 2.5 *If $M \xrightarrow[\mathsf{SLin}]{}\!\!\!\!\!\rightarrow A$ then there exists some separated linear lambda answer A_0 such that $M \xmapsto[\mathsf{SLin}]{}\!\!\!\!\!\rightarrow A_0$.*

We present the proof of this result in Appendix B, which relies on the notion of marking redexes in Appendix A.

2.5 Translations into SLin

Figures 11 and 12 present the adaptations of the Girard translations to the separated calculus. The translations behave exactly as described in previous work [12, 20]: the standard translation allows any function's argument to be duplicated or discarded, while the steadfast translation allows any value to be duplicated or discarded.

Theorem 1 *Let $M, N \in \mathsf{Name}$.*

(i) $(M[x := N])^\circ \equiv M^\circ[x := N^\circ]$.

(ii) $,\ \vdash_{\mathsf{Name}} M : A$ *if and only if* $^{\blacktriangleright_i}, {}^\circ \vdash_{\mathsf{SLin}} M^\circ : A^\circ$.

Separated steadfast translation

Of types

$$\begin{aligned}A^* &\equiv !^c!^w A^+ \\ (A \to B)^+ &\equiv (A^*) \multimap (B^*) \\ Z^+ &\equiv Z\end{aligned}$$

Of terms

$$\begin{aligned}V^* &\equiv !^c!^w V^+ \\ (M\ N)^* &\equiv (\text{let } !^i z = M^* \text{ in } z)\ N^* \\ (\text{let } x = M \text{ in } N)^* &\equiv \text{let } !^i x = M^* \text{ in } N^* \\ x^+ &\equiv x \\ (\lambda x.\, M)^+ &\equiv \lambda y.\, \text{let } !^i x = y \text{ in } M^*\end{aligned}$$

Of typing environments

$$(,\, , x : A)^* \equiv\ ,^*, x : A^+$$

Figure 12: The separated Girard translations into SLin.

(iii) $M \xrightarrow[\mathsf{Name}]{}\!\!\!\!\!\rightarrow N$ *if and only if* $M^\circ \xrightarrow[\mathsf{SLin}]{}\!\!\!\!\!\rightarrow N^\circ$.

(iv) $M \xmapsto[\mathsf{Name}]{}\!\!\!\!\!\rightarrow N$ *if and only if* $M^\circ \xmapsto[\mathsf{SLin}]{}\!\!\!\!\!\rightarrow N^\circ$.

Theorem 2 *Let $M, N \in \mathsf{Val}$.*

(i) $(M[x := V])^* \equiv M^*[x := V^+]$.

(ii) $,\ \vdash_{\mathsf{Val}} M : A$ *if and only if* $^{\blacktriangleright_i}, {}^* \vdash_{\mathsf{SLin}} M^* : A^*$, *and* $,\ \vdash_{\mathsf{Val}} V : A$ *if and only if* $^{\blacktriangleright_i}, {}^* \vdash_{\mathsf{SLin}} V^+ : A^+$.

(iii) *If* $M \xrightarrow[\mathsf{Val}]{}\!\!\!\!\!\rightarrow N$ *then* $M^* \xrightarrow[\mathsf{SLin}]{}\!\!\!\!\!\rightarrow N^*$.

(iv) $M \xmapsto[\mathsf{Val}]{}\!\!\!\!\!\rightarrow N$ *if and only if* $M^* \xmapsto[\mathsf{SLin}]{}\!\!\!\!\!\rightarrow N^*$.

The proofs of these two results as well as the counterexample to completeness in Clause (iii) of Theorem 2 are essentially the same as with the unitary target [20].

Finally, Figure 13 describes the hybrid translation motivated in the introduction. We translate all values to be copyable but not necessarily discardable, and all function arguments to be discardable, but not necessarily copyable. So a call-by-need function mapping the type A to the type B is translated to a copyable linear function from the image of A made explicitly discardable, into the image of B: that is, into the type $!^c((!^w A^\circledast) \multimap B^\circledast)$.

We now begin a series of lemmas which lead to Theorem 3, the soundness and completeness result for the hybrid transformation. We will use the grammar and mapping of Figure 14, which we characterize by the following lemmas:

Lemma 2.6 *Let $M \in \mathsf{Need}$ and $M^\circledast \xrightarrow[\mathsf{SLin}]{}\!\!\!\!\!\rightarrow N$. Then N matches some S in the grammar of Figure 14. Moreover if and $M^\circledast \xmapsto[\mathsf{SLin}]{}\!\!\!\!\!\rightarrow N$, N matches some $\mathcal{S}$ in that grammar.*

Lemma 2.7 *Let $M \in \mathsf{Need}$. Then $(M^\circledast)^\natural \equiv M$.*

Syntactic domains

Evaluation contexts	E	$::=$	$[\,] \mid E\,M$
			$\mid$ let $!^c x = E$ in $M \mid$ let $!^w x = E$ in M
			$\mid$ let $!^i x = E$ in $M \mid$ let $!^i x = !^c E$ in M
			$\mid$ let $!^i x = !^w M$ in $E \mid$ let $!^i x = !^w E$ in $E[x]$
Answers	A	$::=$	$\lambda x.\, M \mid !^w M \mid !^c M \mid$ let $!^i x = !^w M$ in A

Reduction rules

$(\beta\multimap)$	$(\lambda x.\, M)\, N$	$\xrightarrow[\mathsf{SLin}]{}$	$M[x := N]$
$(\beta!^c)$	let $!^c x = !^c N$ in M	$\xrightarrow[\mathsf{SLin}]{}$	$M[x := N]$
$(\beta!^w)$	let $!^w x = !^w N$ in M	$\xrightarrow[\mathsf{SLin}]{}$	$M[x := N]$
$(\beta!^i_{cw})$	let $!^i x = !^c!^w N$ in M	$\xrightarrow[\mathsf{SLin}]{}$	$M[x := N]$
$(\beta!^i_{wc})$	let $!^i x = !^w!^c N$ in $E[x]$	$\xrightarrow[\mathsf{SLin}]{}$	$(E[x])[x := N]$
$(!^i\multimap)$	(let $!^i x = !^w M$ in A) N	$\xrightarrow[\mathsf{SLin}]{}$	let $!^i x = !^w M$ in $(A\, N)$
$(!^\Box!^i)$	let $!^\Box x =$ (let $!^i y = !^w M$ in A) in N where $!^\Box$ ranges over $!^w$, $!^c$, $!^i$.	$\xrightarrow[\mathsf{SLin}]{}$	let $!^i y = !^w M$ in (let $!^\Box x = A$ in N)
$(!^i!^w!^w)$	let $!^i x = !^w$(let $!^i y = !^w M$ in A) in $E[x]$	$\xrightarrow[\mathsf{SLin}]{}$	let $!^i y = !^w M$ in let $!^i x = !^w A$ in $E[x]$
$(!^i!^c!^w)$	let $!^i x = !^c$(let $!^i y = !^w M$ in A) in N	$\xrightarrow[\mathsf{SLin}]{}$	let $!^i y = !^w M$ in let $!^i x = !^c A$ in N

Figure 10: Evaluation in SLin.

Hybrid translation

Of types

$$
\begin{aligned}
A^{\circledast} &\equiv\ !^c A^{\oplus}\\
(A \to B)^{\oplus} &\equiv\ (!^w A^{\circledast}) \multimap B^{\circledast}\\
Z^{\oplus} &\equiv\ Z
\end{aligned}
$$

Of terms

$$
\begin{aligned}
V^{\circledast} &\equiv\ !^c V^{\oplus}\\
(M\ N)^{\circledast} &\equiv\ (\text{let } !^c z = M^{\circledast} \text{ in } z)\ (!^w N^{\circledast})\\
(\text{let } x = M \text{ in } N)^{\circledast} &\equiv\ \text{let } !^i x = !^w M^{\circledast} \text{ in } N^{\circledast}\\
x^{\oplus} &\equiv\ x\\
(\lambda x.\, M)^{\oplus} &\equiv\ \lambda y.\, \text{let } !^i x = y \text{ in } M^{\circledast}
\end{aligned}
$$

Of typing environments

$$(,\,, x : A)^{\circledast} \equiv\ ,^{\circledast}, x : A^{\oplus}$$

Figure 13: The hybrid translation from Need into SLin.

Lemma 2.8 *Every* SLin *answer and evaluation context formable from terms $\mathcal{S}$ are given by $\mathcal{A}$ and $\mathcal{E}$, respectively; every* SLin *evaluation context formable from terms $\mathcal{P}$ is given by $\mathcal{G}$.*

Lemma 2.9 *For all U and $\mathcal{U}$, $U^{\bowtie}$ and $\mathcal{U}^{\bowtie}$ are values in* Need*; for all $\mathcal{A}$, $\mathcal{A}^{\bowtie}$ is an answer in* Need*; for all $\mathcal{E}$ and $\mathcal{G}$, $\mathcal{E}^{\natural}$ and $\mathcal{G}^{\natural}$ are evaluation contexts in* Need*.*

Proof: The above results are all clear, in some cases with the obvious structural induction, from an easy inspection of the definitions. ♦

Lemma 2.10 *Let $\mathcal{E} \vdash_{\mathsf{SLin}} S :!^c \bar{A}$ and $\mathcal{F}, {}^{\blacktriangleright_c} x : \bar{A} \vdash_{\mathsf{SLin}} D[x] : \bar{B}$. Then $(\mathcal{E}, \mathcal{F})^{\natural} \vdash_{\mathsf{Need}} D^{\natural}[S^{\natural}] : \bar{B}^{\natural}$.*

Proof: Since $x \notin \mathrm{fv}(D)$ we have $\mathcal{E} \vdash_{\mathsf{SLin}} \text{let } !^c z = S \text{ in } z : \bar{A}$; then by Lemma 2.1, $\mathcal{E}, \mathcal{F} \vdash_{\mathsf{SLin}} D[\text{let } !^c z = S \text{ in } z] : \bar{B}$. The result is immediate as $(D[\text{let } !^c z = S \text{ in } z])^{\bowtie} \equiv D^{\bowtie}[S^{\natural}]$. ♦

Lemma 2.11 *Let D, S, T be as in Figure 14. Then:*

$$D^{\bowtie}[\text{let } x = S^{\natural} \text{ in } T^{\natural}] \xrightarrow[(C)]{}\!\!\!\!\rightarrow \text{let } x = S^{\natural} \text{ in } D^{\bowtie}[T^{\natural}]\ .$$

Proof: By induction on D. ♦

Under the preceding lemmas, completeness for $\circledast$ is implied by the following soundness result for $\natural$:

Lemma 2.12 *Let S, T, U and so forth be as in Figure 14.*

(i) $(S[x := U])^{\natural} \equiv S^{\natural}[x := U^{\bowtie}]$, $(P[x := U])^{\bowtie} \equiv P^{\bowtie}[x := U^{\bowtie}]$ *and* $(D[y][x := U])^{\bowtie} \equiv (D^{\bowtie}[y])[x := U^{\bowtie}]$.

Reduction-closed images of Need types and terms

$$
\begin{array}{llll}
\text{Types} & A, B & ::= & !^c \bar{A} \\
& \bar{A}, \bar{B} & ::= & (!^w A) \multimap B \mid Z \\
\text{Terms} & S, T & ::= & !^c U \mid P\ (!^w S) \\
& & \mid & \text{let } !^i x = !^w S \text{ in } T \\
& & \mid & \text{let } !^c x = S \text{ in } (D[x]\ !^w T)\ ,\ x \notin \text{fv}(D, T) \\
\text{Values} & U & ::= & x \mid \lambda y.\text{let } !^i x = y \text{ in } S \\
& P & ::= & U \mid \text{let } !^i x = !^w S \text{ in } P \\
& & \mid & \text{let } !^c x = S \text{ in } D[x]\ ,\quad x \notin \text{fv}(D) \\
& D & ::= & [\,] \mid D\ (!^w S)
\end{array}
$$

Evaluation-closed images of Need types and terms

$$
\begin{array}{llll}
\text{Terms} & \mathcal{S}, \mathcal{T} & ::= & !^c \mathcal{U} \mid \mathcal{P}\ (!^w \mathcal{S}) \\
& & \mid & \text{let } !^i x = !^w \mathcal{S} \text{ in } \mathcal{T} \\
\text{Values} & \mathcal{U} & ::= & x \mid \lambda y.\text{let } !^i x = y \text{ in } \mathcal{S} \\
& \mathcal{P} & ::= & \mathcal{U} \mid \text{let } !^i x = !^w \mathcal{S} \text{ in } \mathcal{P} \\
& & \mid & \text{let } !^c x = \mathcal{S} \text{ in } x \\
\text{Answers} & \mathcal{A} & ::= & !^c \lambda y.\text{let } !^i x = y \text{ in } \mathcal{S} \\
& & \mid & \text{let } !^i x = !^w \mathcal{S} \text{ in } \mathcal{A} \\
S\text{-Evaluation Contexts} & \mathcal{E} & ::= & [\,] \mid !^c \mathcal{G} \mid \mathcal{G}\ (!^w S) \\
& & \mid & \text{let } !^i x = !^w S \text{ in } \mathcal{E} \\
& & \mid & \text{let } !^i x = !^w \mathcal{E} \text{ in } \mathcal{E}[x] \\
P\text{-Evaluation Contexts} & \mathcal{G} & ::= & [\,] \mid \text{let } !^i x = !^w S \text{ in } \mathcal{G} \\
& & \mid & \text{let } !^i x = !^w \mathcal{E} \text{ in } \mathcal{G}[x] \\
& & \mid & \text{let } !^c x = \mathcal{E} \text{ in } x
\end{array}
$$

The inverse translation

Of types

$$
\begin{array}{rcl}
(!^c \bar{A})^\natural & \equiv & \bar{A}^\bowtie \\
((!^w A) \multimap B)^\bowtie & \equiv & A^\natural \to B^\natural \\
Z^\bowtie & \equiv & Z
\end{array}
$$

Of terms

$$
\begin{array}{rcl}
(!^c U)^\natural & \equiv & U^\natural \\
(P\ (!^w S))^\natural & \equiv & P^\bowtie\ S^\natural \\
(\text{let } !^i x = S \text{ in } T)^\natural & \equiv & \text{let } x = S^\natural \text{ in } T^\natural \\
(\text{let } !^c x = S \text{ in } (D[x])\ T)^\natural & \equiv & (D^\bowtie[S^\natural])\ T^\natural \\
x^\bowtie & \equiv & x \\
(\lambda y.\text{let } !^i x = y \text{ in } S)^\bowtie & \equiv & \lambda x.S^\natural \\
(\text{let } !^c x = S \text{ in } D[x])^\bowtie & \equiv & D^\bowtie[S^\natural] \\
(\text{let } !^i x = S \text{ in } P)^\bowtie & \equiv & \text{let } !^i x = S^\natural \text{ in } P^\bowtie \\
[\,]^\bowtie & \equiv & [\,] \\
(D\ (!^w S))^\bowtie & \equiv & D^\bowtie\ S^\natural
\end{array}
$$

Of typing environments

$$
\begin{array}{rcl}
(,\ ,x : \bar{A})^\natural & \equiv & ,\ ^\natural, x : \bar{A}^\bowtie \\
(^{\blacktriangleright_i},\ ,\ ^{\blacktriangleright_w}\Delta, {}^{\blacktriangleright_c}\Phi, {}^{\blacktriangleright_l}\Psi)^\natural & \equiv & ,\ ^\natural, \Delta^\natural, \Phi^\natural, \Psi^\natural
\end{array}
$$

Figure 14: The inverse hybrid translation from SLin to Need.

(ii) If $\mathcal{E} \vdash_{\mathsf{SLin}} S : A$ *then* $\mathcal{E}^\natural \vdash_{\mathsf{Need}} S^\natural : A^\natural$,
if $\mathcal{E} \vdash_{\mathsf{SLin}} P : \bar{A}$ *then* $\mathcal{E}^\natural \vdash_{\mathsf{Need}} P^\bowtie : \bar{A}^\bowtie$,
if $\mathcal{E} \vdash_{\mathsf{SLin}} U : \bar{A}$ *then* $\mathcal{E}^\natural \vdash_{\mathsf{Need}} U^\bowtie : \bar{A}^\bowtie$ *and*
if $\mathcal{E} \vdash_{\mathsf{SLin}} D[x] : \bar{A}$ *then* $\mathcal{E}^\natural \vdash_{\mathsf{Need}} D[x]^\bowtie : \bar{A}^\bowtie$.

(iii) If $S \xrightarrow[\mathsf{SLin}]{}\!\!\!\!\rightarrow\ T$ *then* $S^\natural \xrightarrow[\mathsf{Need}]{}\!\!\!\!\rightarrow\ T^\natural$.

(iv) If $S \xmapsto[\mathsf{SLin}]{}\!\!\!\!\rightarrow\ T$ *then* $S^\natural \xmapsto[\mathsf{Need}]{}\!\!\!\!\rightarrow\ T^\natural$.

Proof: Clause (i) is straightforward; note that we may in fact have $x \equiv y$ in the final equivalence.

For Clause (ii), we proceed by structural induction on the depth of the typing proof. The reasoning for the image of the structural SLin rules for all four syntactic groups is the same, and is largely trivial; we write out that reasoning once in Figure 15. We present the reasoning for terms S, P and U in Figures 16, 17 and 18, respectively. For terms of the form $D[x]$, where $D \equiv [\,]$ we have just the Id rule for Need; for $D \equiv D_0\ (!^w S)$ we have an induction as in $\multimap$-E in Figure 16, replacing P with $D[x]$. In many of these cases we may have structural rules mingled within the base trees in the figures, for example in a typing proof of a term S which ends in a $\multimap$-E step we might have

$$
\dfrac{\mathcal{E} \vdash P : (!^w A) \multimap B \qquad \dfrac{\dfrac{\dfrac{\vdots\ \mathcal{P}_2}{^{\blacktriangleright_i},\ ,\ ^{\blacktriangleright_w}\Delta \vdash S : A}\ !^w\text{-I}}{^{\blacktriangleright_i},\ ,\ ^{\blacktriangleright_w}\Delta \vdash\ !^w S : !^w A}\ \text{Wkning}}{^{\blacktriangleright_i},\ ,\ ^{\blacktriangleright_w}\Delta, x : A_0 \vdash\ !^w S : !^w A}\ \vdots\ \mathcal{P}_1}{\mathcal{E}, {}^{\blacktriangleright_i},\ ,\ ^{\blacktriangleright_w}\Delta, x : A_0 \vdash P\ (!^w S) : B}\ \multimap\text{-E}\ ,
$$

but it is clear that these rules can simply be shifted out of the way; in this example we have simply

$$
\dfrac{\mathcal{E} \vdash P : (!^w A) \multimap B \qquad \dfrac{\dfrac{\dfrac{\vdots\ \mathcal{P}_2}{^{\blacktriangleright_i},\ ,\ ^{\blacktriangleright_w}\Delta \vdash S : A}\ \text{Wkning}}{^{\blacktriangleright_i},\ ,\ ^{\blacktriangleright_w}\Delta, x : A_0 \vdash S : A}\ !^w\text{-I}}{^{\blacktriangleright_i},\ ,\ ^{\blacktriangleright_w}\Delta, x : A_0 \vdash\ !^w S : !^w A}\ \vdots\ \mathcal{P}_1}{\mathcal{E}, {}^{\blacktriangleright_i},\ ,\ ^{\blacktriangleright_w}\Delta, x : A_0 \vdash P\ (!^w S) : B}\ \multimap\text{-E}\ .
$$

For Clause (iii), we have soundness for top level reduction of S terms as shown in Figure 19, and top-level P terms as shown in Figure 20. D and U terms are reducible only within their S- and P-subterms. Compatible closure is straightforward as the translations depend only on the top-level structure of terms, and not their subterms.

For Clause (iv), we have soundness with the preceding lemmas about preservation of answers and evaluation contexts as shown in Figures 21 and 22; compatible closure follows again by the preservation lemmas. ⧫

Lemma 2.13 *Let A be a call-by-need answer and E be a call-by-need evaluation context. Then $A^\circledast$ is a separated-linear answer, and if we take $[\,]^\circledast \equiv [\,]$ then $E^\circledast$ is a separated-linear evaluation context.*

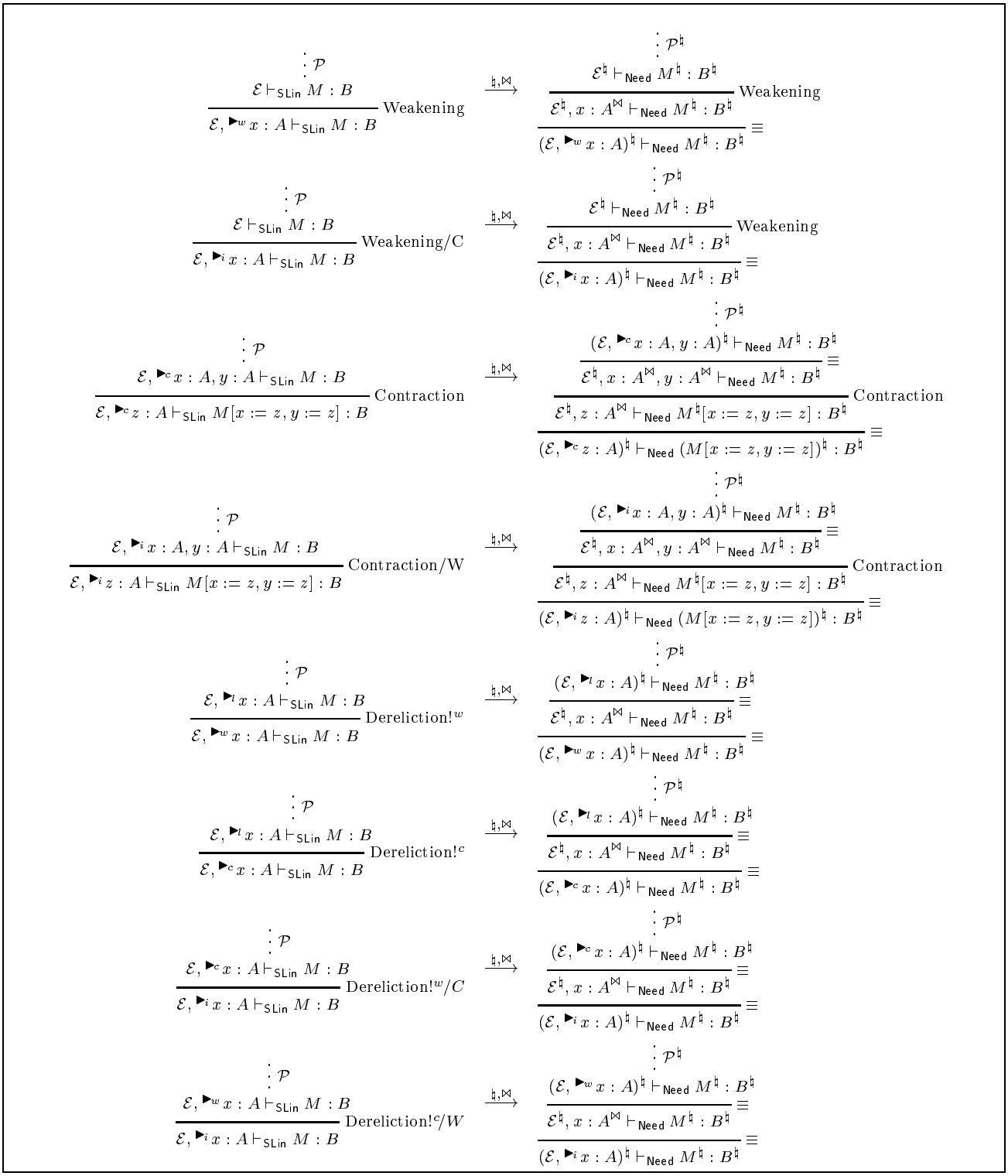


Figure 15: Soundness of $\natural$ for (completeness of $\circledast$ from) structural typing rules. We slightly abuse our metavariable notation by writing M, N to range over any of the subsets S, U and so forth of SLin; we moreover handwave a bit by not writing out explicitly that the terms and types on the right-hand side of the type inference statements of the trees on the right-hand side of the arrows could also be translated with $\bowtie$.

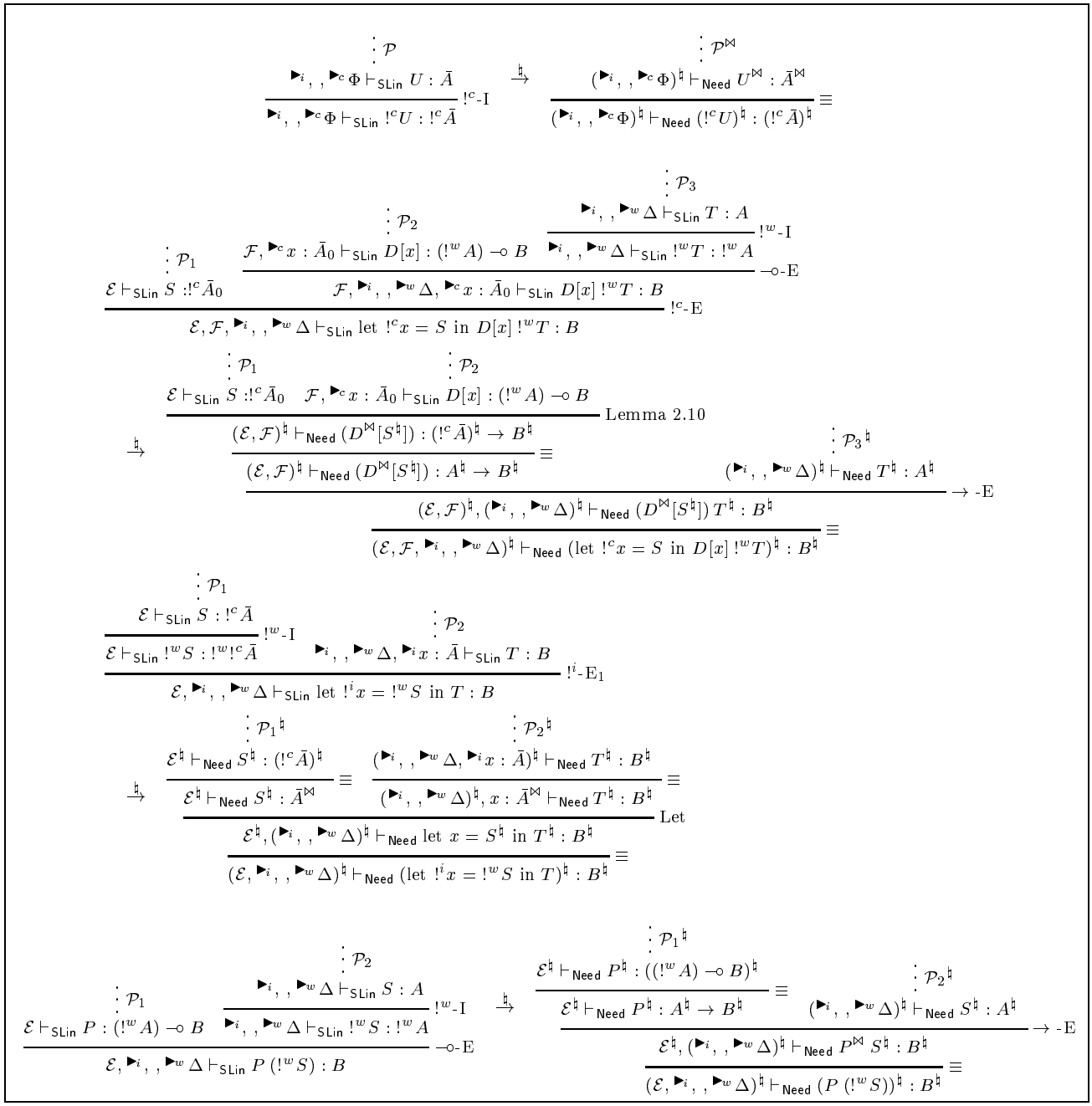


Figure 16: Soundness of $\natural$ for (completeness of $\circledast$ from) typing judgments for terms S.

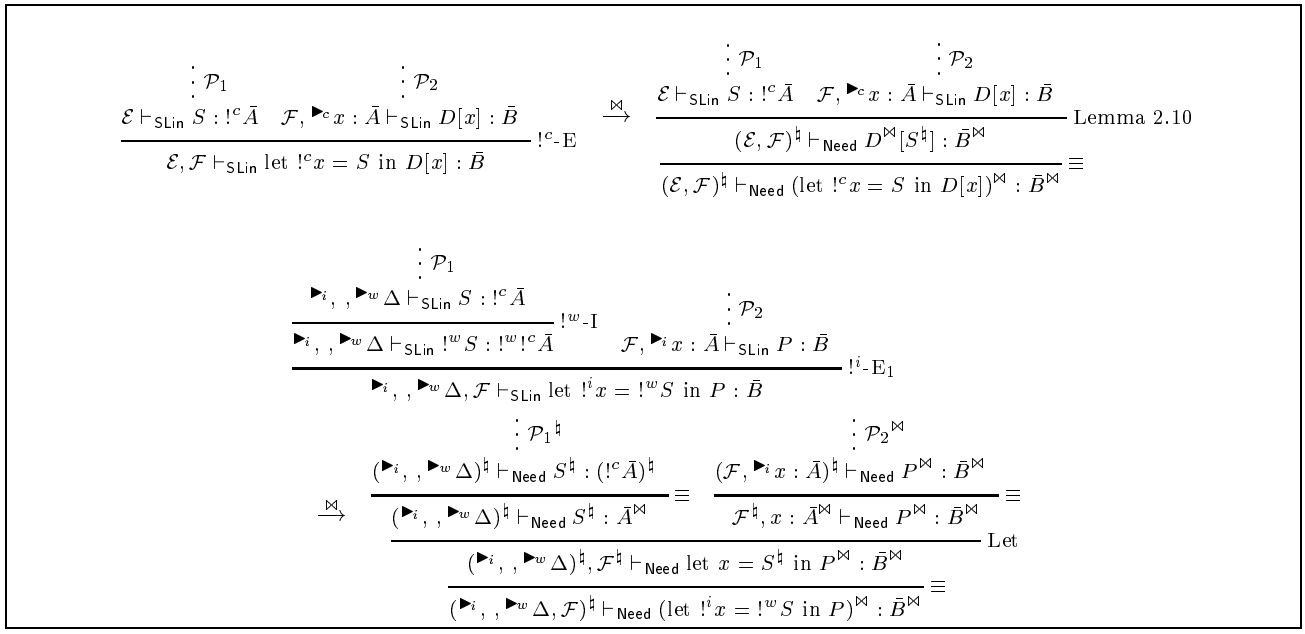


Figure 17: Soundness of $\bowtie$ for (completeness of $\natural$ from) typing judgments for terms P.

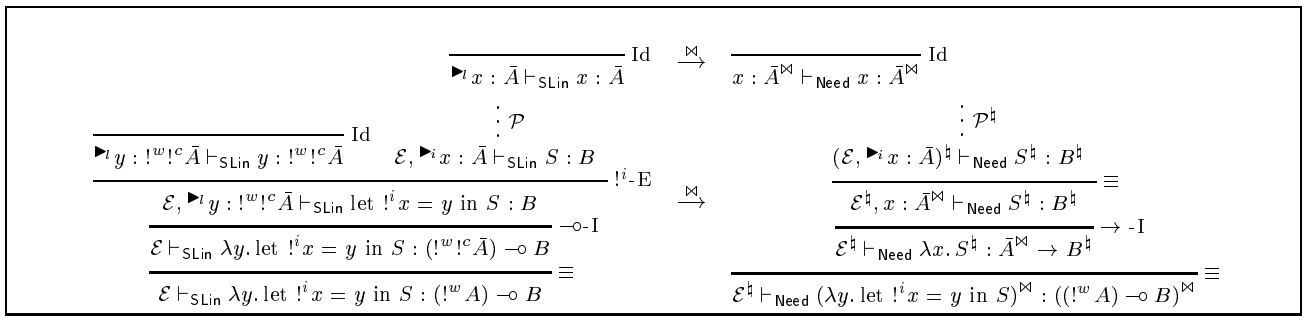


Figure 18: Soundness of $\bowtie$ for (completeness of $\oplus$ for) typing judgments for terms U.

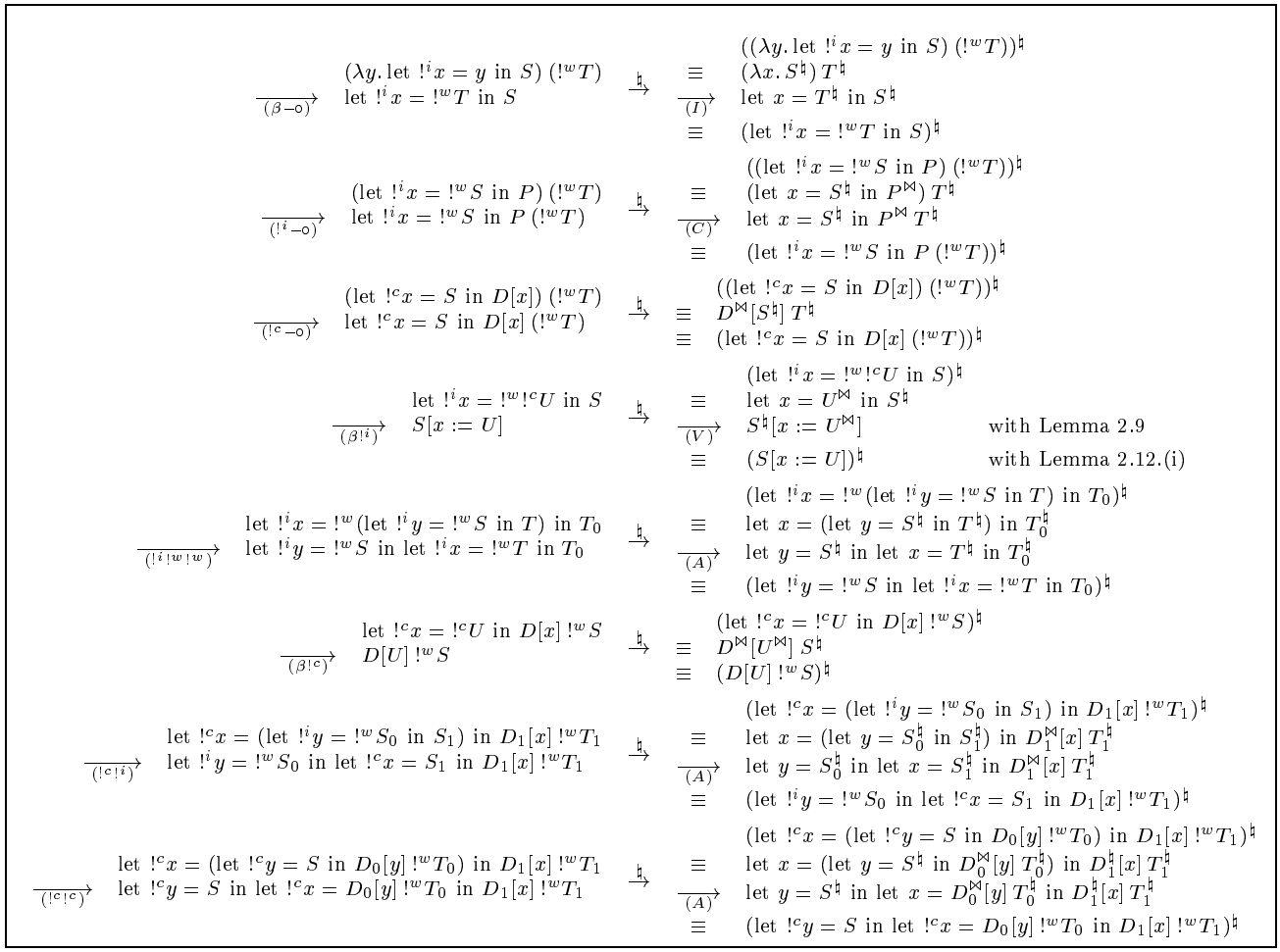


Figure 19: Soundness of $\natural$ for reduction of terms S.

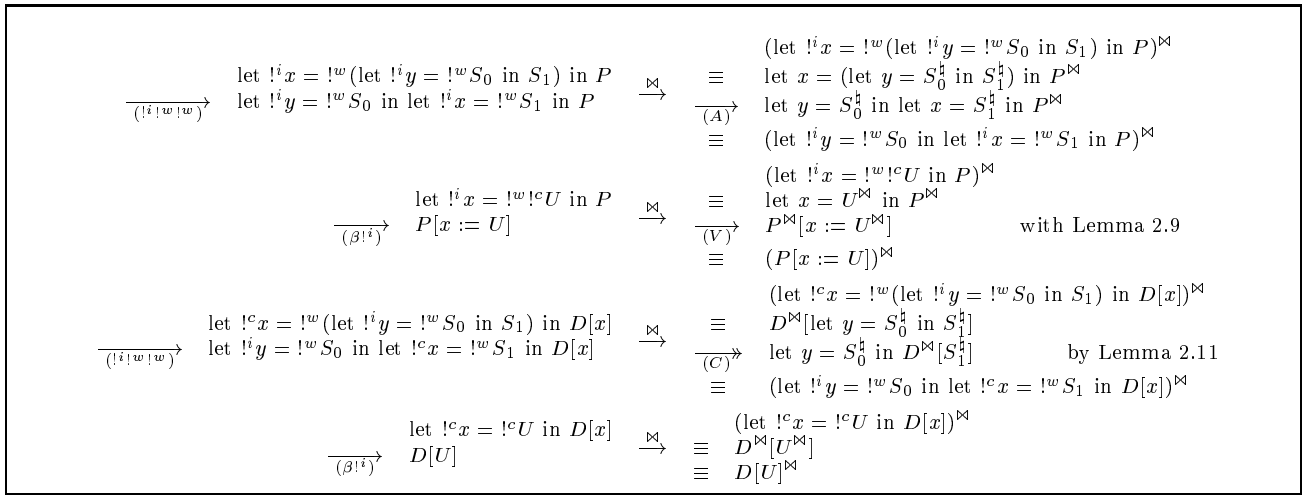


Figure 20: Soundness of $\bowtie$ for reduction of terms P.

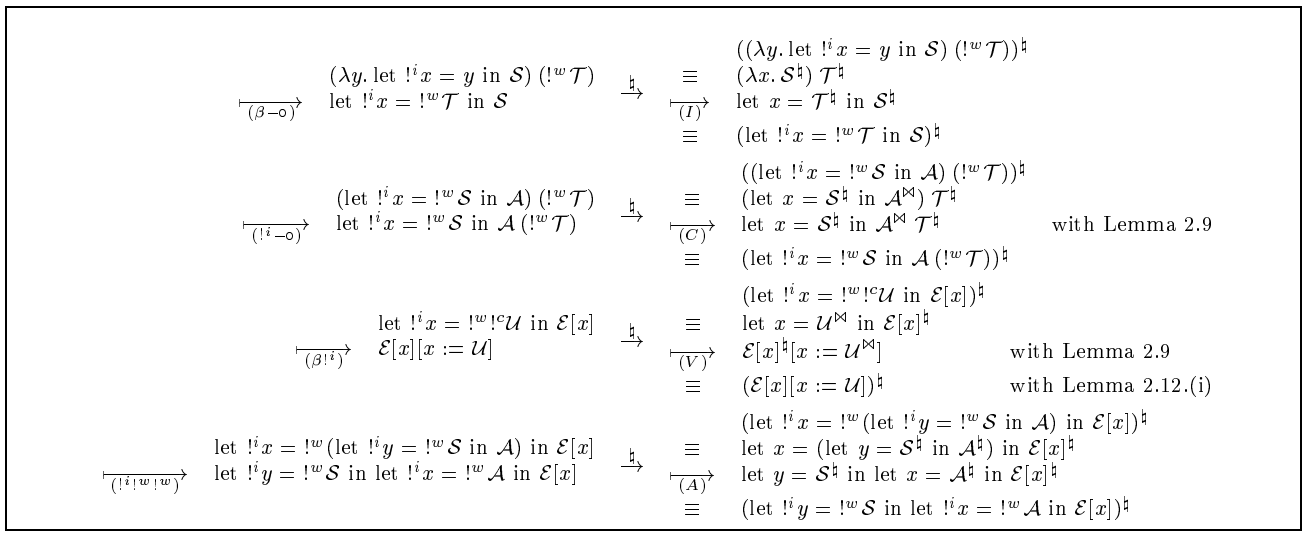


Figure 21: Soundness of $\natural$ for evaluation of terms S.

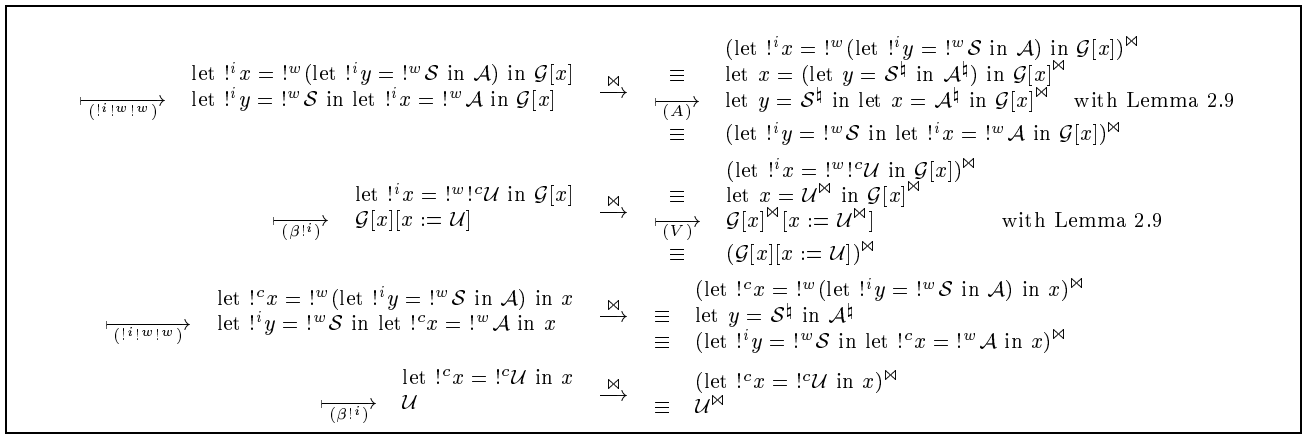


Figure 22: Soundness of $\bowtie$ for evaluation of terms P.

Proof: Simple, by induction on the size of the structures. ♦

Corollary 2.14 *Let E be a call-by-need evaluation context. Then there exists a separated-linear evaluation context E_0 such that for all x we have $(E[x])^{\circledast} \equiv E_0[x]$.*

Proof: This E_0 is simply $E^{\circledast}[!^c[\,]]$. ♦

Theorem 3 *Let $M, N \in \mathsf{Need}$.*

(i) $(M[x := V])^{\circledast} \equiv M^{\circledast}[x := V^{\oplus}]$.

(ii) $, \vdash_{\mathsf{Need}} M : A$ *if and only if* $^{\blacktriangleright_i}, {}^{\circledast} \vdash_{\mathsf{SLin}} M^{\circledast} : A^{\circledast}$, *and* $, \vdash_{\mathsf{Need}} V : A$ *if and only if* $^{\blacktriangleright_i}, {}^{\circledast} \vdash_{\mathsf{SLin}} V^{\oplus} : A^{\oplus}$.

(iii) $M \xrightarrow[\mathsf{Need}]{}\!\!\!\!\rightarrow N$ *if and only if* $M^{\circledast} \xrightarrow[\mathsf{SLin}]{}\!\!\!\!\rightarrow N^{\circledast}$.

(iv) $M \xmapsto[\mathsf{Need}]{}\!\!\!\!\rightarrow N$ *if and only if* $M^{\circledast} \xmapsto[\mathsf{SLin}]{}\!\!\!\!\rightarrow N^{\circledast}$.

Proof: Clause (i) is trivial since all values are translated with the same prefix. Soundness in Clause (ii) follows by a straightforward induction on the depth of the proof tree; we illustrate the reasoning for each Need typing rule in Figure 23. Soundness in Clause (iii) follows by a similar analysis of each rule, shown in Figure 24. Finally, soundness for Clause (iv) follows by the analyses of each rule in Figure 25, Lemma 2.13 and Corollary 2.14.

For completeness we rely on the subset of SLin and mapping from it to Need given in Figure 14. It is easy to verify that this grammar includes all $\circledast$-images of Need terms and is closed under general SLin reduction, that the mapping $\natural$ is right-inverse to $\circledast$, and that $\natural$ is sound for typing proofs and reduction sequences, which together imply completeness in Clauses (ii) and (iii). Finally, completeness in Clause (iv) follows since $\natural$ takes SLin answers and evaluation contexts to Need answers and evaluation contexts as well. ♦

Why is $\circledast$ complete for general reduction from Need into SLin while $*$ from Val into Lin or from Need into Aff, which produce similar reduction-closed images of terms, are not? The counterexample for $*$ exploits the fact that we cannot distinguish an eliminator introduced by the translation for massaging the left-hand side of an application from an eliminator which is the translation of a let-binding or an abstraction. Under $\circledast$ we do not have this confusion: the former is translated as a $!^c$-eliminator while the latter becomes a $!^i$-eliminator, which can be distinguished in the inverse translation.

3 Conclusion

The separated-linear lambda calculus uses a more refined control of the substructural operations on types and terms than previous systems based on linear logic. The additional control allows call-by-need to be incorporated in a much more satisfying manner within the scope of the Girard translation, by a new transformation, into the same system as call-by-name and call-by-value, and in a way which is both sound and complete for reduction. We end this report with a sketch of some variations, extensions and application of the separated calculus.

Omitting the structural typing rules. It is also possible to formulate the typing rules for SLin without the Dereliction, Weakening and Contraction rules. Rather than a single Id rules and four Dereliction rules, we would have one axiom link per zone; rather than explicit weakening and contraction rules, we would make the different abilities of different zones implicit in other rules. We present the typing relation $\vdash_{\mathsf{SRF}}$ in Figure 26 (structural rule-free). This formulation is similar to Barber's presentation of dual intuitionistic linear logic [4].

Permission to weaken variables in certain zones in implicit in the axiom rules. For example in Id_l,

$$\frac{}{^{\blacktriangleright_i}, , {}^{\blacktriangleright_w}\Delta, {}^{\blacktriangleright_l} x : A \vdash_{\mathsf{SRF}} x : A}\ \mathrm{Id}_l$$

the variables in , and Δ are quite clearly unused, which is acceptable since we allow weakening in those two zones. On the other hand, no unused assumptions appear in the zones where weakening is not allowed. Permission to contract is implicit in the elimination rules with two inferences above the bar. For example in $\multimap$-E, we can use variables in , and Φ to type the same variables in both subterms of the application, which is acceptable since contraction is allowed in those two zones. On the other hand, in the other zones we must use separate environments for the different subterms.

Although we have fewer $\vdash_{\mathsf{SRF}}$ typing rules in total, they are individually more complicated than the corresponding $\vdash_{\mathsf{SLin}}$ rules. Our shorthand with $\mathcal{E}$, $^{\blacktriangleright_l}$, etc. is not so useful here since the pairwise interactions of zones is more important and must be made explicit. Nonetheless, it is clear that the schemes produce judgments of the same types, and we believe that the shorter, less intricate rules of $\vdash_{\mathsf{SLin}}$ allow simpler, easier to read proofs, even if a few are longer.

However it should be noted that the removal of structural rules does allow a refinement of the Curry-Howard correspondence. Since under $\vdash_{\mathsf{SLin}}$ we can rearrange the location of structural rules within a proof fairly arbitrarily without changing a term, several proofs of a term's type can be given. In other words, each well-typed term can be Curry-Howard correspondent with many proofs in the logic. Omitting the structural rules, we in fact have an isomorphism between closed terms and theorems (To see that isomorphism fails for open terms in general one need only note that all four Id rules in $\vdash_{\mathsf{SRF}}$ prove the same thing).

Larger logical issues. We have implictly raised but not answered the question of what the decomposition of ! into $!^w$ and $!^c$ does to the original system of linear logic. A complete answer to that question is clearly beyond the scope of this report, where we take the logic to be in service of the calculi. Nonetheless, we do believe that a single-sided sequent calculus for a separated-linear logic should be straightforward, although we have not yet attempted to verify its relation to the models of Girard [12] or Jacobs [15].

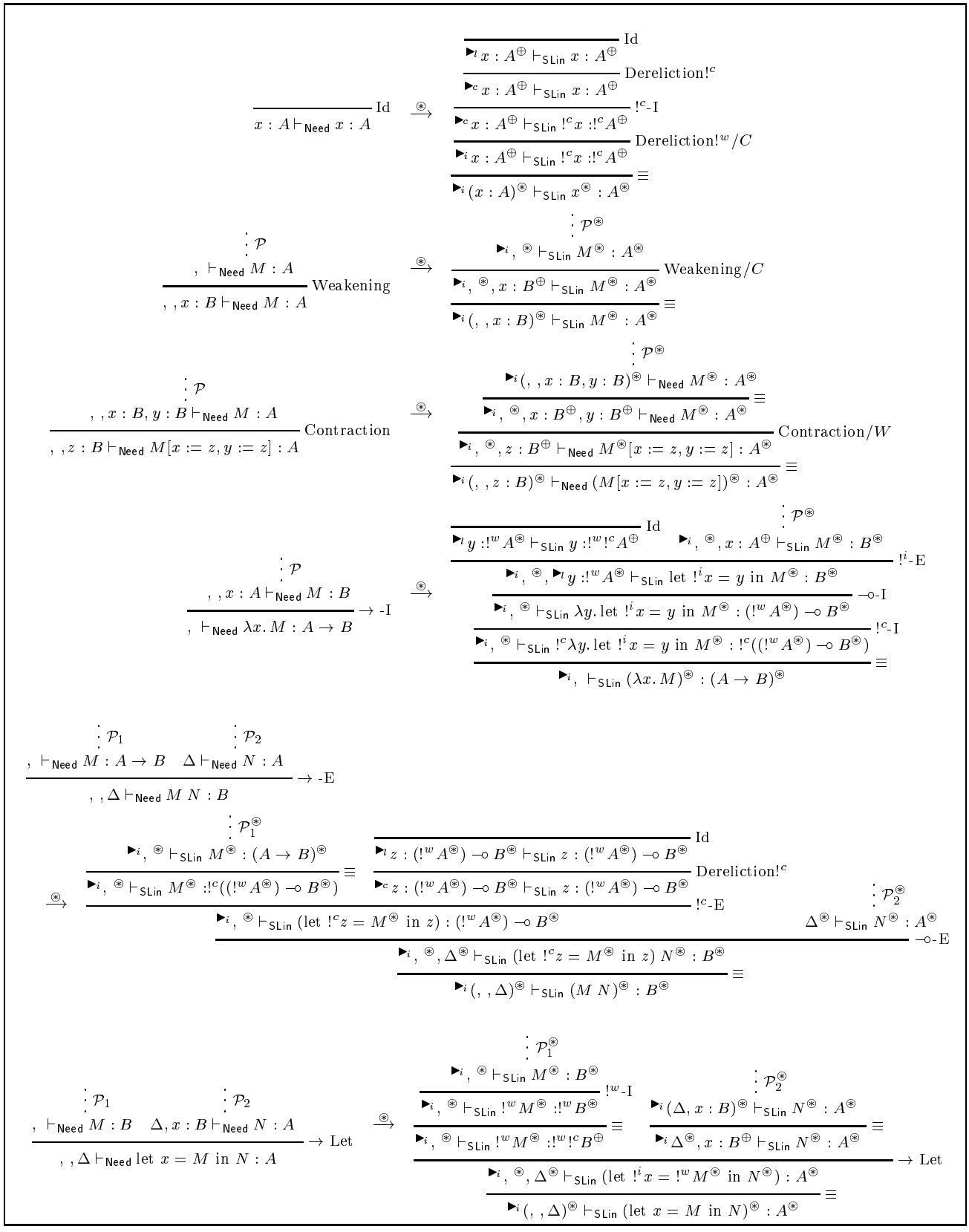


Figure 23: Soundness of $\circledast$ for types.

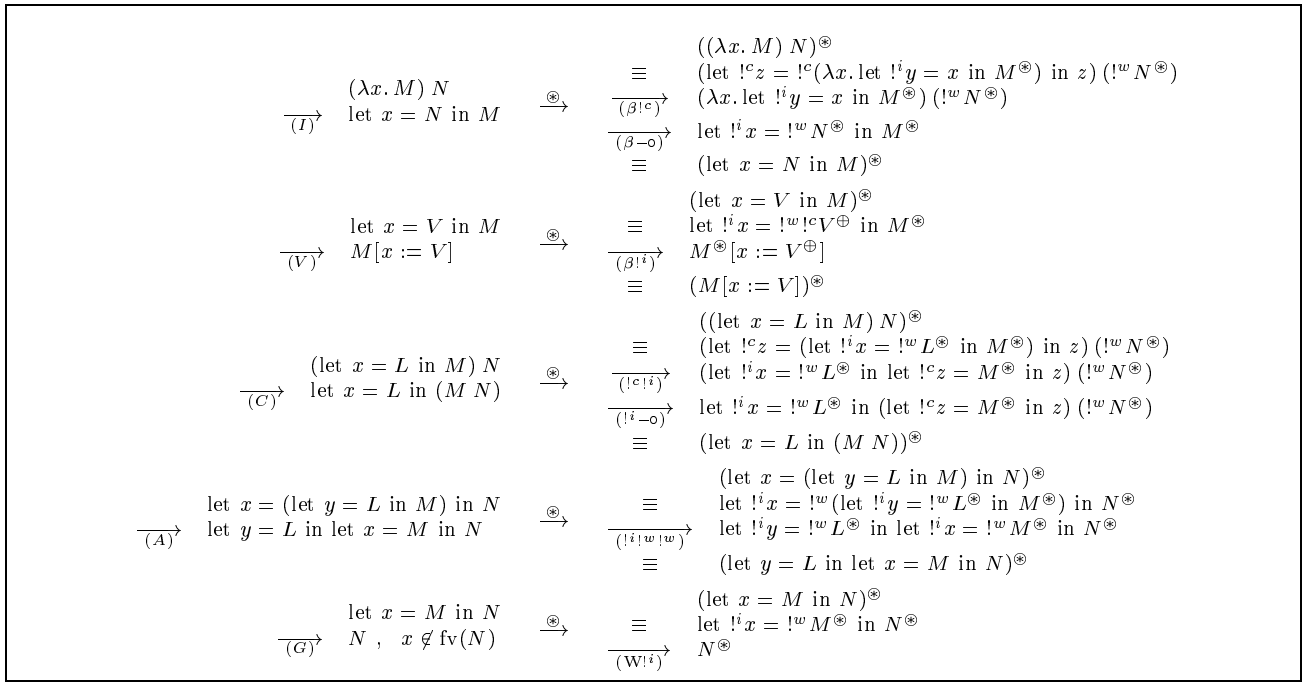


Figure 24: Soundness of $\circledast$ for general reduction.

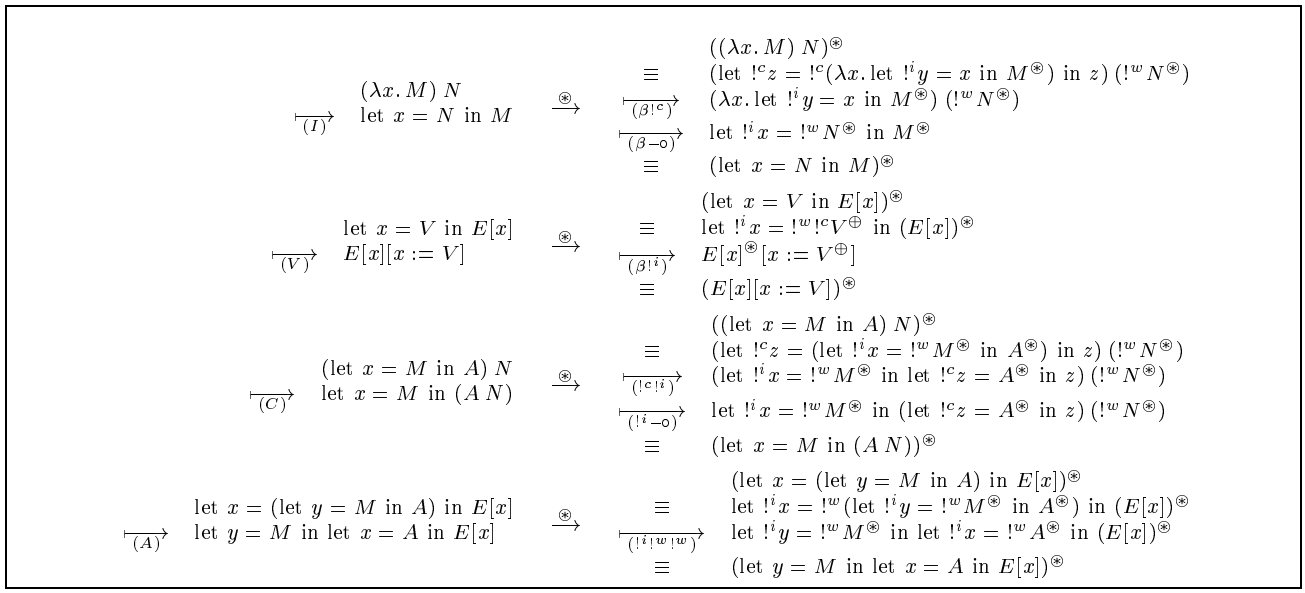


Figure 25: Soundness of $\circledast$ for top-level standard reduction. Lemma 2.13 and Corollary 2.14 guarantee the applicability of the standard SLin rules.

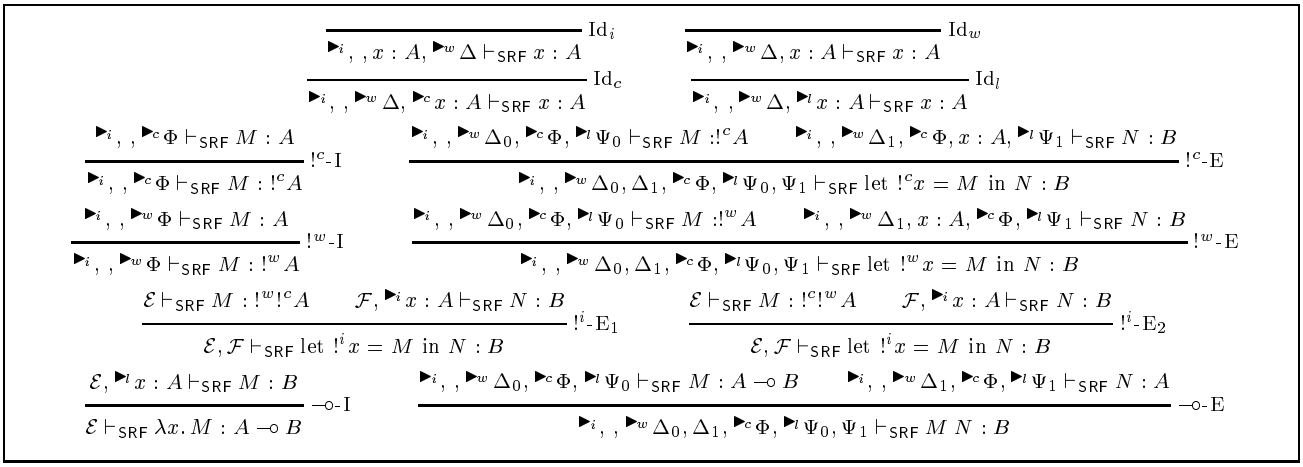


Figure 26: The structural rule-free typing relation $\vdash_{\mathsf{SRF}}$ for SLin.

Product and sum types. Generally as a basis for programming languages one would want to consider a larger calculus, in particular with product and sum types. An extension to the standard translation to map both sums and products from a suitable extension of call-by-name, into the linear lambda calculus extended with the multiplicative and additive sum connectives (& and $\oplus$, in Girard's formulation), is straightforward. However, the extension of the steadfast translation is less clear. While the extension for products is merely quite awkward, it is not at all clear how to achieve the extension for sums.

One can envision an extension of the hybrid translation in the manner of either original translation. For the standard translation, the results are straightforward, while an attempt in the style of the steadfast traslation inherits the difficulties of the original. We present these issues in more detail elsewhere [18], and do not pursue them further here.

Acknowledgments.

I'd like to thank Martin Odersky, David N. Turner, Philip Wadler, Martin Wehr and a number of anonymous referees for their comments through the development of this work.

References

[1] S. Abramsky, Computational interpretations of linear logic, *Theoretical Computer Science* **111** (1993) 3–57.

[2] Z. M. Ariola, M. Felleisen, J. Maraist, M. Odersky and P. Wadler, A call-by-need lambda calculus, in: *Conf. Rec. POPL'95: 22nd ACM Symp. on Principles of Programming Languages* (ACM Press, San Francisco, California, January 1995) 233–246.

[3] Z. M. Ariola and M. Felleisen, The call-by-need lambda calculus, in: (Technical Report CIS-TR-94-23, Department of Computer Science, University of Oregon, October 1994; extended version submitted to: *J. Functional Programming*).

[4] A. Barber, DILL — dual intuitionistic linear logic (to appear; draft paper, Dual intuitionistic linear logic, October 1995).

[5] H. P. Barendregt, *The Lambda Calculus: Its Syntax and Semantics*, volume 103 of *Studies in Logic and the Foundations of Computer Science* (North-Holland Publishing Company, 1981).

[6] P. N. Benton, G. Bierman, V. de Paiva and M. Hyland, Type assignment for intuitionistic linear logic (Technical Report 262, Computing Laboratory, University of Cambridge, August 1992).

[7] P. N. Benton, Strong normalisation for the linear term calculus, *Journal of Functional Programming* **5**:1 (January 1995) 65–80.

[8] G. Bierman, What is a categorical model of intuitionistic linear logic?, in: *Proc. Second International Conference on Typed Lambda Calculus* (Springer Verlag, Lecture Notes in Computer Science 902, Edinburgh, Scotland, April 1995).

[9] G. Bierman, *On Intuitionistic Linear Logic* (Ph.D. Thesis, University of Cambridge Computer Laboratory, December 1993; also appears as Technical Report 346, Computing Laboratory, University of Cambridge, August 1994).

[10] A. Church, *The Calculi of Lambda Conversion* (Princeton University Press, 1941).

[11] H. B. Curry and R. Feys, *Combinatory Logic*, vol. 1 (North-Holland, Amsterdam, 1958).

[12] J.-Y. Girard, Linear logic, *Theoretical Computer Science* **50** (1987) 1–102.

[13] J.-Y. Girard, On the unity of logic, *Annals of Pure and Applied Logic* **59** (1993) 201–217.

[14] W. Howard, The formulae-as-types notion of construction, in: *To H. B. Curry: Essays on Combinatory Logic, Lambda-Calculus and Formalism* (Academic Press, 1980).

[15] B. Jacobs, Semantics of lambda-I and of other substructure lambda calculi, in: *Typed Lambda Calculus and Applications* (Springer LNCS 664, 1994) 195-208.

[16] B. Jacobs, Semantics of weakening and contraction, *Annals of Pure and Applied Logic* **69** (1994) 73–106.

[17] I. Mackie, *The Geometry of Implementation* (Ph.D. Thesis, Imperial College London, 1994).

[18] J. Maraist, *Comparing Reduction Strategies in Resource-Conscious Lambda Calculi* (Doctoral thesis, University of Karlsruhe, to appear Winter 1996/7).

[19] J. Maraist, M. Odersky and P. Wadler, The call-by-need lambda calculus (Unabridged) (October 1994, Technical Reports 28/94, Fakultät für Informatik, Universität Karlsruhe and FP-94-11, Department of Computing Science, University of Glasgow; extended version submitted to: *J. Functional Programming*).

[20] J. Maraist, M. Odersky, D. N. Turner and P. Wadler, Call-by-name, call-by-value, call-by-need and the linear lambda calculus, in: *Proc. MFPS'95: Eleventh Conf. on the Mathematical Foundations of Programming Semantics* (Elsevier Publishers, ENTCS 1, New Orleans, March/April 1995).

[21] G. D. Plotkin, Call-by-name, call-by-value and the λ calculus, *Theoretical Computer Science* **1** (1975) 125–159.

[22] J. C. Reynolds, The discoveries of continuations, *Lisp and Symbolic Computation* **6**:3/4 (November 1993) 233–248.

[23] S. R. della Rocca and L. Roversi, Lambda calculus and intuitionistic linear logic (Manuscript, July 1994). Available from L. Roversi, University of Pisa, rover@di.unipi.it.

[24] P. Wadler, Linear types can change the world!, in: M. Broy and C. Jones, eds., *Programming Concepts and Methods* (North Holland, Sea of Galilee, Israel, April 1990).

[25] P. Wadler, A syntax for linear logic, in: *Proc. MFPS'93: Ninth Int. Conf. on the Mathematical Foundations of Programming Semantics* (Springer Verlag, Lecture Notes in Computer Science 802, New Orleans, Louisiana, April 1993).

A Marked Separated-Linear Reduction

Figure 27 describes $\mathsf{SLin}^\star$, a variation of the SLin reduction rules. $\mathsf{SLin}^\star$ and SLin share a common term language, but $\mathsf{SLin}^\star$ allows single $(!{\multimap}, !!)$ steps to move more than one eliminator prefix at a time. As a result, single-step $\mathsf{SLin}^\star$ reduction relates more terms than single-step SLin reduction, although their reflexive, transitive closures are clearly equal:

Lemma A.1 *Let $M, N \in \mathsf{SLin}$. We have $M \xrightarrow[\mathsf{SLin}]{}\!\!\!\!\!\rightarrow N$ if and only if $M \xrightarrow[\mathsf{SLin}^\star]{}\!\!\!\!\!\rightarrow N$.*

Figure 28 shows the syntax and Figure 29 the reduction rules of the marked variant $\mathsf{SLin}^{\star\prime}$ of $\mathsf{SLin}^\star$. Compatible closure in $\mathsf{SLin}^{\star\prime}$ is more complicated than usual, and we present these closure rules in Figure 30. We refer to reduction by rules $(\beta{\multimap}_0, \beta!^\Box_0, \star!^\Box{\multimap}_0, \star!^\Box{\multimap}^<_0, \star!^\Box!^\triangle_0, \star!^\Box!^{\triangle^<}_0, !^i!^\Box!^w_0, !^i!^\Box!^{w^<}_0)$ as ${\mathsf{SLin}^{\star\prime}}_0$. We have the usual correspondence between the marked and unmarked systems.

Definition 1 *Let $M' \in \mathsf{SLin}^{\star\prime}$; then we take $|M'|$ to be the $\mathsf{SLin}^\star$ term formed simply by erasing all markings in M'.*

Lemma A.2 *Let $\sigma' : M' \xrightarrow[\mathsf{SLin}^{\star\prime}]{}\!\!\!\!\!\rightarrow N'$. Then $|M'| \xrightarrow[\mathsf{SLin}^\star]{}\!\!\!\!\!\rightarrow |N'|$.*

Proof: Trivial, since every rule of $\mathsf{SLin}^\star$ can be retrieved by simply erasing the markings. ♦

In the above lemma, we refer to the sequence $|M'| \xrightarrow[\mathsf{SLin}^\star]{}\!\!\!\!\!\rightarrow |N'|$ as $|\sigma'|$, and define projection of multistep reduction sequences accordingly as the concatenation of the projections of the single steps.

Lemma A.3 *Let $M' \in \mathsf{SLin}^{\star\prime}$ and $\sigma : |M'| \xrightarrow[\mathsf{SLin}^\star]{}\!\!\!\!\!\rightarrow N$. Then there exists some $N' \in \mathsf{SLin}^{\star\prime}$, $N \equiv |N'|$, such that $\sigma' : M' \xrightarrow[\mathsf{SLin}^{\star\prime}]{}\!\!\!\!\!\rightarrow N'$ and $\sigma \equiv |\sigma'|$.*

Proof: Again, immediate from the correspondence of the rules in the calculi. ♦

It will be useful to have a more specific lemma about ${\mathsf{SLin}^{\star\prime}}_0$ reduction.

Lemma A.4 *Let $L', M', N' \in {\mathsf{SLin}^{\star\prime}}_0$, with $M' \xrightarrow[{\mathsf{SLin}^{\star\prime}}_0]{}\!\!\!\!\!\rightarrow N'$. Then:*

- *$M'[x := L'] \xrightarrow[{\mathsf{SLin}^{\star\prime}}_0]{}\!\!\!\!\!\rightarrow N'[x := L']$.*
- *$L'[x := M'] \xrightarrow[{\mathsf{SLin}^{\star\prime}}_0]{}\!\!\!\!\!\rightarrow L'[x := N']$.*

Proof: Again the first result is by compatible closure, and the second by structural induction. ♦

As usual, there is a correspondence between marked terms and sets of references to redexes of the corresponding unmarked term. A *path* is a finite, possibly empty sequence of symbols referring to subterms of a term structure; a path is *valid* for a term if it actually corresponds to that term's structure. We leave the symbols largely unspecified; 0 and 1 will suffice. For example, in the term $M \equiv (\lambda x.\, M_0)\, M_1$, if we take 0 to mean the left- and 1 the right-subterm (or just 0 for terms with only one child), then "0" is a valid path referring to (indexing) the term $\lambda x.\, M_0$, "00" indexes the term M_0, and "1" indexes the term M_1; "01" is not a valid path for M. For a term M, we let $\mathsf{p}(M)$ be the set of valid paths into M, and let γ, δ range over paths.

We index $(\beta, !!!)$ redex by a single path, and $(!{\multimap}, !!)$ redexes by a pair of a path and a positive integer:

Definition 2 *An* indexing set *$\mathcal{F}$ for a term $M \in \mathsf{SLin}^\star$ is a subset of $\mathsf{p}(M) \cup (\mathsf{p}(M) \times \mathbb{N})$ where:*

1. *If $\gamma \in \mathcal{F}$, then γ indexes a $(\beta, !!!)$ redex in M.*
2. *If $(\gamma, n) \in \mathcal{F}$, then γ indexes a $(!{\multimap}, !!)$ redex in M which moves n eliminator prefixes.*
3. *If $(\gamma, n_0), (\gamma, n_1) \in \mathcal{F}$, then $n_0 = n_1$.*

It is clear from the definition of the rules that redexes specified by different rules will never coincide at the top-level: that is, if $(\gamma_0, n_0), \gamma_1 \in \mathcal{F}$, then $\gamma_0 \not\equiv \gamma_1$. Clause (3) of the definition allows us to make the claim that indexing sets and marked terms are in one-to-one correspondence; without that restriction we would be able to mark certain terms as more than one redex. The following lemma is then clear:

Lemma A.5 *If $\mathcal{F}$ be an indexing set for M, then there exists some marked term M' where $|M'| \equiv M$ and every redex indexed in $\mathcal{F}$ is marked in M'. If M' is a marked term, then there exists an indexing set $\mathcal{F}$ for $|M'|$ such that every redex marked in M' is indexed.*

Syntactic domains As in SLin, noting as well:

$$\text{Prefixes} \quad P \quad ::= \text{let } !^i x = M \text{ in } \mid \text{let } !^w x = M \text{ in } \mid \text{let } !^c x = M \text{ in}$$

Main reduction rules As in SLin.

(!–○) rules

$$(\star !^{\square} \text{–}\circ) \qquad (P^n \text{ let } !^{\square} x = L \text{ in } M)\ N \xrightarrow[\mathsf{SLin}^{\star}]{} P^n \text{ let } !^{\square} x = L \text{ in } (M\ N)$$
where $!^{\square}$ ranges over $!^w$, $!^c$, $!^i$.

(!!) rules

$$(\star !^{\square} !^{\triangle}) \qquad \text{let } !^{\square} x = (P^n \text{ let } !^{\triangle} y = L \text{ in } M) \text{ in } N \xrightarrow[\mathsf{SLin}^{\star}]{} P^n \text{ let } !^{\triangle} y = L \text{ in (let } !^{\square} x = M \text{ in } N)$$
where $!^{\square}$ and $!^{\triangle}$ each range over $!^w$, $!^c$, $!^i$.

(!!!) rules As in SLin.

$$(!^i !^{\square} !^w) \qquad \text{let } !^i x = !^{\square}(\text{let } !^i y = !^w L \text{ in } M) \text{ in } N \xrightarrow[\mathsf{SLin}]{} \text{let } !^i y = !^w L \text{ in let } !^i x = !^{\square} M \text{ in } N$$
where $!^{\square}$ ranges over $!^w$ and $!^c$.

Displacement function

$$\begin{aligned}
\mathbf{d}(\sigma, 0) &= 0 \\
\mathbf{d}(\sigma : M \xrightarrow[(\beta,\mathrm{W})]{} N, m) &= -1,\ m > 0 \text{ and top-level } \sigma \\
\mathbf{d}(\sigma : M \xrightarrow[(!!!)]{} N, m) &= 1,\ m > 0 \text{ and top-level } \sigma \\
\mathbf{d}(\sigma : M \xrightarrow[(!\text{–}\circ)]{} N, m) &= 0 \\
\mathbf{d}(\sigma : M \xrightarrow[(!!)]{} N, m) &= n,\ m > 0,\ \text{top-level } \sigma \text{ and } n \text{ prefixes moved} \\
\mathbf{d}(P\ M \to P\ N, m) &= \mathbf{d}(M \to N, m-1),\ m > 0 \\
\mathbf{d}(M \to N, m) &= 0,\ \text{other compatible closure}
\end{aligned}$$

Figure 27: The generalized separated linear lambda calculus.

So it is sensible to write $(|M'|, \mathcal{F}) \equiv M'$ for the appropriate indexing set, and we adopt this notation henceforward.

The restriction of Clause (3) also means that we cannot track every generalized redex in a term simultaneously, but since we are interested in $\mathsf{SLin}^{\star}$ only for the insight it allows into SLin, such a restriction is acceptable.

A.1 Developments and their Finiteness

We define residuals and developments as usual. Let $\sigma' : (|M'|, \mathcal{F}) \equiv M' \xrightarrow[\mathsf{SLin}^{\star\prime}]{} N'$; then there exists some $\mathcal{G}$ such that $N' \equiv (|N'|, \mathcal{G})$. Then we define $\mathcal{G}$ to be the *residual(s)* of the redexes in $\mathcal{F}$ with respect to σ'. A *development* of a marked term is a reduction sequence which contracts only marked redexes, *viz.* a $\mathsf{SLin}^{\star\prime}_0$ sequence. A development is *finite* if it has a finite number of steps, and a finite development is *complete* if its final result is unmarked. So to show that all developments are finite (Corollary A.14), we will show that $\mathsf{SLin}^{\star\prime}_0$ is strongly normalizing (Lemma A.13).

As usual, for strong normalization of $\mathsf{SLin}^{\star\prime}_0$, we use a decoration of the marked calculus with weights on variable occurrences. We refer to the weighted calculus as $\mathsf{SLin}^{\star\bullet}$ and present the norm $\|\,.\,\| : \mathsf{SLin}^{\star\bullet} \to \mathbb{N}$ in Figure 31. We take $\dot{M}, \dot{N}$ and so forth to range over $\mathsf{SLin}^{\star\bullet}$ terms. We take the same reduction rules in $\mathsf{SLin}^{\star\bullet}$ as

$$\begin{aligned}
\|x\|^i &= i \\
\|\lambda x.\dot{M}\| &= \|\dot{M}\| \\
\|\dot{M}\ \dot{N}\| &= 2\|\dot{M}\| + 2\|\dot{N}\| \\
\|\dot{P}\ \dot{N}\| &= 2\|\dot{P}\| + \|\dot{N}\| \\
\|\text{let } !^{\square} x = \dot{M} \text{ in}\| &= \|\dot{M}\| \\
\|!^{\square}\dot{M}\| &= \|\dot{M}\|
\end{aligned}$$

Figure 31: The norm $\|\,.\,\|$ on $\mathsf{SLin}^{\star\bullet}$ terms.

in $\mathsf{SLin}^{\star\prime}$, as refer to $\mathsf{SLin}^{\star\bullet}_0$ as for $\mathsf{SLin}^{\star\prime}_0$. Once again, we have the usual substitution results.

Lemma A.6 *Let $\dot{L}, \dot{M}, \dot{N} \in \mathsf{SLin}^{\star\bullet}$, with $\dot{M} \xrightarrow[\mathsf{SLin}^{\star\bullet}]{} \dot{N}$. Then:*

- $\dot{M}[x := \dot{L}] \xrightarrow[\mathsf{SLin}^{\star\bullet}]{} \dot{N}[x := \dot{L}]$.
- $\dot{L}[x := \dot{M}] \xrightarrow[\mathsf{SLin}^{\star\bullet}]{}\!\!\!\!\!\rightarrow \dot{L}[x := \dot{N}]$.

Proof: The first result is again just compatible closure, and the second follows by induction on the structure of $\dot{L}$. ♦

Just as we can separate a marking from a term, we can separate weights from a marked term. Given a term

Syntactic domains

Terms	L', M', N'	$::= x \mid \lambda x.\, M' \mid !^w M' \mid !^c M' \mid M'\ N' \mid \mathcal{P}\ M$
		$\mid {}^{[\beta \multimap \circ]}(\lambda x.\, M')\ N'$
		$\mid {}^{[\star^{!^c} \multimap \circ_n]}(\mathcal{P}^{n-1}\ {}^{c}\mathcal{P}\ M')\ N'$
		$\mid {}^{[\star^{!^w} \multimap \circ_n]}(\mathcal{P}^{n-1}\ {}^{w}\mathcal{P}\ M')\ N'$
		$\mid {}^{[\star^{!^i} \multimap \circ_n]}(\mathcal{P}^{n-1}\ {}^{i}\mathcal{P}\ M')\ N'$
General eliminator prefixes	$\mathcal{P}$	$::= {}^{i}\mathcal{P} \mid {}^{w}\mathcal{P} \mid {}^{c}\mathcal{P}$
$!^i$-eliminator prefixes	${}^{i}\mathcal{P}$	$::=$ let $!^i x = M'$ in
		$\mid {}^{i^w}\mathcal{P} \mid {}^{i^c}\mathcal{P}$
		$\mid {}^{[\star^{!^i} \mid^w {}_n]}$let $!^i x = (\mathcal{P}^{n-1}\ {}^{w}\mathcal{P}\ M')$ in
		$\mid {}^{[\star^{!^i} \mid^c {}_n]}$let $!^i x = (\mathcal{P}^{n-1}\ {}^{c}\mathcal{P}\ M')$ in
		$\mid {}^{[\star^{!^i} \mid^i {}_n]}$let $!^i x = (\mathcal{P}^{n-1}\ {}^{i}\mathcal{P}\ M')$ in
– with a $!^w$-prefixed bound term	${}^{i^w}\mathcal{P}$	$::=$ let $!^i y = !^w L'$ in
		$\mid {}^{[\beta^{!^i}_{cw}]}$let $!^i x = !^w !^c M'$ in
		$\mid {}^{[!^i \mid^w !^w]}$let $!^i x = !^w({}^{i^w}\mathcal{P}\ M')$ in
– with a $!^c$-prefixed bound term	${}^{i^c}\mathcal{P}$	$::=$ let $!^i y = !^c L'$ in
		$\mid {}^{[\beta^{!^i}_{wc}]}$let $!^i x = !^c !^w M'$ in
		$\mid {}^{[!^i \mid^c !^w]}$let $!^i x = !^c({}^{i^w}\mathcal{P}\ M')$ in
$!^c$-eliminator prefixes	${}^{c}\mathcal{P}$	$::=$ let $!^c x = M'$ in $\mid {}^{\mathcal{E}}\mathcal{P}$
		$\mid {}^{[\star^{!^c} \mid^w {}_n]}$let $!^c x = (\mathcal{P}^{n-1}\ {}^{w}\mathcal{P}\ M')$ in
		$\mid {}^{[\star^{!^c} \mid^c {}_n]}$let $!^c x = (\mathcal{P}^{n-1}\ {}^{c}\mathcal{P}\ M')$ in
		$\mid {}^{[\star^{!^c} \mid^i {}_n]}$let $!^c x = (\mathcal{P}^{n-1}\ {}^{i}\mathcal{P}\ M')$ in
– with a $!^c$-prefixed bound term	${}^{\mathcal{E}}\mathcal{P}$	$::=$ let $!^c y = !^c L'$ in
		$\mid {}^{[\beta^{!^c}]}$let $!^c x = !^c M'$ in
$!^w$-eliminator prefixes	${}^{w}\mathcal{P}$	$::=$ let $!^w x = M'$ in $\mid {}^{w^w}\mathcal{P}$
		$\mid {}^{[\star^{!^w} \mid^w {}_n]}$let $!^w x = (\mathcal{P}^{n-1}\ {}^{w}\mathcal{P}\ M')$ in
		$\mid {}^{[\star^{!^w} \mid^c {}_n]}$let $!^w x = (\mathcal{P}^{n-1}\ {}^{c}\mathcal{P}\ M')$ in
		$\mid {}^{[\star^{!^w} \mid^i {}_n]}$let $!^w x = (\mathcal{P}^{n-1}\ {}^{i}\mathcal{P}\ M')$ in
– with a $!^w$-prefixed bound term	${}^{w^w}\mathcal{P}$	$::=$ let $!^w y = !^w L'$ in
		$\mid {}^{[\beta^{!^w}]}$let $!^w x = !^w M'$ in

Figure 28: Syntax of the marked generalized separated-linear lambda calculus.

Main reduction rules

$$
\begin{array}{lrcl}
(\beta{\multimap}_0) & {}^{[\beta\multimap]}(\lambda x.\, M')\ N' & \xrightarrow[\mathsf{SLin}^{\star\prime}]{} & M'[x := N'] \\
(\beta{\multimap}_1) & (\lambda x.\, M')\ N' & \xrightarrow[\mathsf{SLin}^{\star\prime}]{} & M'[x := N'] \\
(\beta!^c_0) & {}^{[\beta!^c]}\text{let } !^c x = !^c M' \text{ in } N' & \xrightarrow[\mathsf{SLin}^{\star\prime}]{} & N'[x := M'] \\
(\beta!^c_1) & \text{let } !^c x = !^c M' \text{ in } N' & \xrightarrow[\mathsf{SLin}^{\star\prime}]{} & N'[x := M'] \\
(\beta!^w_0) & {}^{[\beta!^w]}\text{let } !^w x = !^w M' \text{ in } N' & \xrightarrow[\mathsf{SLin}^{\star\prime}]{} & N'[x := M'] \\
(\beta!^w_1) & \text{let } !^w x = !^w M' \text{ in } N' & \xrightarrow[\mathsf{SLin}^{\star\prime}]{} & N'[x := M'] \\
(\beta!^i_{cw\,0}) & {}^{[\beta!^i_{cw}]}\text{let } !^i x = !^w !^c M' \text{ in } N' & \xrightarrow[\mathsf{SLin}^{\star\prime}]{} & N'[x := M'] \\
(\beta!^i_{cw\,1}) & \text{let } !^i x = !^w !^c M' \text{ in } N' & \xrightarrow[\mathsf{SLin}^{\star\prime}]{} & N'[x := M'] \\
(\beta!^i_{wc\,0}) & {}^{[\beta!^i_{wc}]}\text{let } !^i x = !^c !^w M' \text{ in } N' & \xrightarrow[\mathsf{SLin}^{\star\prime}]{} & N'[x := M'] \\
(\beta!^i_{wc\,1}) & \text{let } !^i x = !^c !^w M' \text{ in } N' & \xrightarrow[\mathsf{SLin}^{\star\prime}]{} & N'[x := M'] \\
(\mathrm{W}!^i_1) & \text{let } !^i x = !^w M' \text{ in } N' & \xrightarrow[\mathsf{SLin}^{\star\prime}]{} & N',\ x \notin \mathrm{fv}(N')
\end{array}
$$

(!–○) rules

$$
\begin{array}{lrcl}
(\star!^\Box{-}\!\circ_0) & {}^{[\star!^\Box-\circ_n]}(\mathcal{P}^{n-1}\,\Box\mathcal{P}\,M')\ N' & \xrightarrow[\mathsf{SLin}^{\star\prime}]{} & \mathcal{P}^{n-1}\,\Box\mathcal{P}\,(M'\ N') \\
(\star!^\Box{-}\!\circ^<_0) & {}^{[\star!^\Box-\circ_n]}(\mathcal{P}^{m}\,M')\ N' & \xrightarrow[\mathsf{SLin}^{\star\prime}]{} & \mathcal{P}^{m}\,({}^{[\star!^\Box-\circ_{n-m}]}M'\ N'), \\
 & & & 0 < m < n \\
(\star!^\Box{-}\!\circ_1) & (\mathcal{P}^{n-1}\,\Box\mathcal{P}\,M')\ N' & \xrightarrow[\mathsf{SLin}^{\star\prime}]{} & \mathcal{P}^{n-1}\,\Box\mathcal{P}\,(M'\ N') \\
(\star!^\Box{-}\!\circ^>_1) & {}^{[\star!^\Box-\circ_n]}(\mathcal{P}^{n-1}\,\Box\mathcal{P}\,\mathcal{R}^m\,\Box\mathcal{R}\,M')\ N' & \xrightarrow[\mathsf{SLin}^{\star\prime}]{} & \mathcal{P}^{n-1}\,\Box\mathcal{P}\,\mathcal{R}^m\,\Box\mathcal{R}\,(M'\ N'), \\
 & & & m \geq 0
\end{array}
$$

where $!^\Box$ ranges over $!^w$, $!^c$, $!^i$
and correspondingly $\Box\mathcal{P}$ to ${}^{\mathrm{w}}\mathcal{P}$, ${}^{\mathrm{c}}\mathcal{P}$, ${}^{\mathrm{i}}\mathcal{P}$ (and $\Box\mathcal{R}$).

(!!) rules

$$
\begin{array}{lrcl}
(\star!^\Box!^\triangle_0) & {}^{[\star!^\Box!^\triangle_n]}\text{let } !^\Box x = (\mathcal{P}^{n-1}\,\Delta\mathcal{P}\,M') \text{ in } N' & \xrightarrow[\mathsf{SLin}^{\star\prime}]{} & \mathcal{P}^{n-1}\,\Delta\mathcal{P}\,(\text{let } !^\Box x = M' \text{ in } N') \\
(\star!^\Box!^{\triangle<}_0) & {}^{[\star!^\Box!^\triangle_n]}\text{let } !^\Box x = (\mathcal{P}^{m}\,M') \text{ in } N' & \xrightarrow[\mathsf{SLin}^{\star\prime}]{} & \mathcal{P}^{m}\,({}^{[\star!^\Box!^\triangle_{n-m}]}\text{let } !^\Box x = M' \text{ in } N'), \\
 & & & \text{where } 0 < m < n \\
(\star!^\Box!^\triangle_1) & \text{let } !^\Box x = (\mathcal{P}^{n-1}\,\Delta\mathcal{P}\,M') \text{ in } N' & \xrightarrow[\mathsf{SLin}^{\star\prime}]{} & \mathcal{P}^{n-1}\,\Delta\mathcal{P}\,(\text{let } !^\Box x = M' \text{ in } N') \\
(\star!^\Box!^{\triangle>}_1) & {}^{[\star!^\Box!^\triangle_n]}\text{let } !^\Box x = (\mathcal{P}^{n-1}\,\Delta\mathcal{P}\,\Box\mathcal{R}^m\,\mathcal{R}\,M') \text{ in } N' & \xrightarrow[\mathsf{SLin}^{\star\prime}]{} & \mathcal{P}^{n-1}\,\Delta\mathcal{P}\,\mathcal{R}^m\,\Box\mathcal{R}\,(\text{let } !^\Box x = M' \text{ in } N'), \\
 & & & \text{where } m \geq 0
\end{array}
$$

where $!^\Box$ and $!^\triangle$ each range over $!^w$, $!^c$, $!^i$
and correspondingly $\Box\mathcal{P}$ to ${}^{\mathrm{w}}\mathcal{P}$, ${}^{\mathrm{c}}\mathcal{P}$, ${}^{\mathrm{i}}\mathcal{P}$ (and $\Box\mathcal{R}$).

(!!!) rules

$$
\begin{array}{lrcl}
(!^i!^\Box!^w{}_0) & {}^{[\star!^i!^\Box!^w{}_n]}\text{let } !^i x = !^\Box({}^{\mathrm{i}}{}^w\mathcal{P}^n\ M') \text{ in } N' & \xrightarrow[\mathsf{SLin}^{\star\prime}]{} & {}^{\mathrm{i}}{}^w\mathcal{P}^n\ (\text{let } !^i x = !^\Box M' \text{ in } N') \\
(!^i!^\Box!^{w<}_0) & {}^{[\star!^i!^\Box!^w{}_n]}\text{let } !^i x = !^\Box({}^{\mathrm{i}}{}^w\mathcal{P}^m\,{}^{\mathrm{i}}{}^w\mathcal{R}^{n-m}\ M') \text{ in } N' & \xrightarrow[\mathsf{SLin}^{\star\prime}]{} & {}^{\mathrm{i}}{}^w\mathcal{P}^m\ ({}^{[\star!^i!^\Box!^w{}_{n-m}]}{}^{\mathrm{i}}{}^w\mathcal{R}^{n-m}\ \text{let } !^i x = !^\Box M' \text{ in } N') \\
 & & & \text{where } 0 < m < n \\
(!^i!^\Box!^w{}_1) & \text{let } !^i x = !^\Box({}^{\mathrm{i}}{}^w\mathcal{P}^n\ M') \text{ in } N' & \xrightarrow[\mathsf{SLin}^{\star\prime}]{} & {}^{\mathrm{i}}{}^w\mathcal{P}^n\ (\text{let } !^i x = !^\Box M' \text{ in } N') \\
(!^i!^\Box!^{w>}_1) & {}^{[\star!^i!^\Box!^w{}_n]}\text{let } !^i x = !^\Box({}^{\mathrm{i}}{}^w\mathcal{P}^n\ {}^{\mathrm{i}}{}^w\mathcal{R}^m\ M') \text{ in } N' & \xrightarrow[\mathsf{SLin}^{\star\prime}]{} & {}^{\mathrm{i}}{}^w\mathcal{P}^n\ {}^{\mathrm{i}}{}^w\mathcal{R}^m\ (\text{let } !^i x = !^\Box M' \text{ in } N') \\
 & & & \text{where } 0 < m
\end{array}
$$

where $!^\Box$ ranges over $!^w$ and $!^c$

Figure 29: $(\beta, !{\multimap}, !!!)$ reduction rules for the marked generalized separated-linear lambda calculus.

Let $M' \xrightarrow[\mathsf{SLin}]{} N'$. Then:

- $\lambda x.\, M' \to \lambda x.\, N'$,
- $!^w M' \xrightarrow[\mathsf{SLin}]{} !^w N'$,
- $!^c M' \xrightarrow[\mathsf{SLin}]{} !^c N'$,
- $L'\ M' \xrightarrow[\mathsf{SLin}]{} L'\ N'$,
- $M'\ L' \xrightarrow[\mathsf{SLin}]{} N'\ L'$,
- $^{[\beta\multimap]}(\lambda x.\, L')\ M' \xrightarrow[\mathsf{SLin}]{} {}^{[\beta\multimap]}(\lambda x.\, L')\ N'$,
- $^{[\star!^{\Box}\multimap_n]}(\mathcal{P}^{n-1}\,{}^{\Box}\mathcal{P}\,L')\ M' \xrightarrow[\mathsf{SLin}]{} {}^{[\star!^{\Box}\multimap_n]}(\mathcal{P}^{n-1}\,{}^{\Box}\mathcal{P}\,L')\ N'$,
  where $!^{\Box}$ ranges over $!^w$, $!^c$, $!^i$ with corresponding ${}^{\Box}\mathcal{P}$.
- $^{[\star!^{\Box_i}\Delta_n]}\mathrm{let}\ !^{\Box}x = (\mathcal{P}^{n-1}\,\Delta\mathcal{P}\,L')\ \mathrm{in}\ M' \xrightarrow[\mathsf{SLin}]{} {}^{[\star!^{\Box_i}\Delta_n]}\mathrm{let}\ !^{\Box}x = (\mathcal{P}^{n-1}\,\Delta\mathcal{P}\,L')\ \mathrm{in}\ N'$,
  and $^{[\star!^{\Box_i}\Delta_n]}\mathrm{let}\ !^{\Box}x = (\mathcal{P}^{n-1}\,\Delta\mathcal{P}\,M')\ \mathrm{in}\ L' \xrightarrow[\mathsf{SLin}]{} {}^{[\star!^{\Box_i}\Delta_n]}\mathrm{let}\ !^{\Box}x = (\mathcal{P}^{n-1}\,\Delta\mathcal{P}\,N')\ \mathrm{in}\ L'$,
  where $!^{\Box}$ and $!^{\Delta}$ each range over $!^w$, $!^c$, $!^i$ with corresponding $\Delta\mathcal{P}$.

Let $\sigma : \mathcal{P}^n\,{}^{\Box}\mathcal{P}\,M' \xrightarrow[\mathsf{SLin}]{} N'$, where $!^{\Box}$ and $!^{\Delta}$ each range over $!^w$, $!^c$, $!^i$ and with corresponding $\Delta\mathcal{P}$:

- $^{[\star!^{\Box}\multimap_n]}(\mathcal{P}^n\,{}^{\Box}\mathcal{P}\,M')\ L' \xrightarrow[\mathsf{SLin}^{\star\prime}]{} {}^{[\star!^{\Box}\multimap_{n+\mathbf{d}(\sigma)}]}(N')\ L'$.
- $^{[\star!^{\Delta}!^{\Box}_n]}\mathrm{let}\ !^{\Delta}x = (\mathcal{P}^n\,{}^{\Box}\mathcal{P}\,M')\ \mathrm{in}\ L' \xrightarrow[\mathsf{SLin}^{\star\prime}]{} {}^{[\star!^{\Delta}!^{\Box}_{n+\mathbf{d}(\sigma)}]}\mathrm{let}\ !^{\Delta}x = N'\ \mathrm{in}\ L'$.

Let $\sigma : {}^{i^w}\mathcal{P}^n\,M' \xrightarrow[\mathsf{SLin}]{} N'$ and let $!^{\Box}$ range over $!^w$, $!^c$:

- $^{[\star!^i!^{\Box}!^w{}_n]}\mathrm{let}\ !^i x = ({}^{i^w}\mathcal{P}^n\,M')\ \mathrm{in}\ L' \xrightarrow[\mathsf{SLin}^{\star\prime}]{} {}^{[\star!^i!^{\Box}!^w{}_{n+\mathbf{d}(\sigma)}]}\mathrm{let}\ !^i x = N'\ \mathrm{in}\ L'$.

Figure 30: Compatible closure in $\mathsf{SLin}^{\star\prime}$.

$M \in \mathsf{SLin}^{\star}$ we define the variable-paths $\Xi(M)$ to be the set

$$\Xi(M) = \{\gamma \in \mathsf{p}(M) : M|_\gamma \equiv x,\ \text{any } x\}\ ,$$

that is, all paths indexing variables. Then a weighting of a marked term $M' \in \mathsf{SLin}^{\star\prime}$ is just any total function I from the variable-paths of $|M'|$ to integers:

$$I : \Xi(|M'|) \to \mathbb{N}\ .$$

The projection and lifting results for $\mathsf{SLin}^{\star\bullet}$ into $\mathsf{SLin}^{\star\prime}$, and by extension $\mathsf{SLin}^{\star}$ as well, are clear. We will need a lemma describing the interaction of substitution and norms for particular terms.

Lemma A.7 *Let $\dot{M}, \dot{N} \in \mathsf{SLin}^{\star\bullet}$ such that for all occurrences of x^i in $\dot{M}$ we have $\|\dot{N}\| < i$. Then $\|\dot{M}[x := \dot{N}]\| \leq \|\dot{M}\|$.*

Proof: Trivially, by induction; equality arises only when x is not free in $\dot{M}$. ♦

We identify a particular class of weightings relevant to marked reduction.

Definition 3 *A term $\dot{M} \in \mathsf{SLin}^{\star\bullet}$ has a* decreasing weighting *if it meets all of the following conditions:*

1. *For every subterm $^{[\beta\multimap]}(\lambda x.\, \dot{M}_0)\ \dot{M}_1$, and for all occurrences of x^i within $\dot{M}_0$, we have $\|\dot{M}_1\| < 1$.*
2. *For every subterm $^{[\beta!^{\Box}]}\mathrm{let}\ !^{\Box}x = !^{\Box}\dot{M}_0\ \mathrm{in}\ \dot{M}_1$ where $!^{\Box}$ ranges over $!^w$,$!^c$, and for all occurrences of x^i within $\dot{M}_1$, we have $\|\dot{M}_0\| < 1$.*
3. *For every subterm $^{[\beta!^i]}\mathrm{let}\ !^i x = !^{\Box}!^{\Delta}\dot{M}_0\ \mathrm{in}\ \dot{M}_1$ with $!^{\Box}$ and $!^{\Delta}$ dual, and for all occurrences of x^i within $\dot{M}_1$, we have $\|\dot{M}_0\| < 1$.*

As suggested by the name, the norm is decreased by $\mathsf{SLin}^{\star\bullet}_0$ reduction of terms with a decreasing weighting. Moreover, $\mathsf{SLin}^{\star\bullet}_0$-reducing a term will again have a decreasing weighting.

Lemma A.8 *Let $\dot{M} \xrightarrow[\mathsf{SLin}^{\star\bullet}_0]{} \dot{N}$, where $\dot{M}$ has a decreasing weighting. Then $\|\dot{M}\| > \|\dot{N}\|$.*

Proof: By a structural induction down to the various possible top-level contractions. We show the base cases for the groups of redexes, and a representative of the trivial induction steps.

1. *Top level marked $(\beta\multimap)$ contraction.* We have $\dot{M} \equiv {}^{[\beta\multimap]}(\lambda x.\, \dot{M}_0)\ \dot{M}_1$ and $\dot{N} \equiv \dot{M}_0[x := \dot{M}_1]$; by the definition of a decreasing weighting we have that the weighting of every occurrence of x^i in $\dot{M}_0$ is greater than $\|\dot{M}_1\|$, which by Lemma A.7 means that $\|\dot{M}_0[x := \dot{M}_1]\| \leq \|\dot{M}_0\|$. Then we have the result with

$$\begin{aligned} & \|{}^{[\beta\multimap]}(\lambda x.\, \dot{M}_0)\ \dot{M}_1\| \\ = \ & 2\|\dot{M}_0\| + 2\|\dot{M}_1\| \\ > \ & \|\dot{M}_0\| \\ \geq \ & \|\dot{M}_0[x := \dot{M}_1]\|\ . \end{aligned}$$

2. *Top level marked* $(\beta!^{\square})$ *contractions,* $!^{\square}$ *ranging over* $!^w, !^c$. We have $\dot{M} \equiv {}^{[\beta!^{\square}]}\text{let } !^{\square}x = !^{\square}\dot{M}_0 \text{ in } \dot{M}_1$ and $\dot{N} \equiv \dot{M}_1[x := \dot{M}_0]$. We can again use Lemma A.7 and the definition of a decreasing weighting to show that $\|\dot{M}_1[x := \dot{M}_0]\| \leq \|\dot{M}_1\|$. Then similarly we have the result with

$$\begin{aligned} & \|^{[\beta!^{\square}]}\text{let } !^{\square}x = !^{\square}\dot{M}_0 \text{ in } \dot{M}_1\| \\ = \;& 2\|\dot{M}_0\| + \|\dot{M}_1\| \\ > \;& \|\dot{M}_1\| \\ \geq \;& \|\dot{M}_1[x := \dot{M}_0]\| \ . \end{aligned}$$

3. *Top level marked* $(\beta!^i)$ *contractions.* We have $\dot{M} \equiv {}^{[\beta!^i]}\text{let } !^i x = !^{\square}!^{\triangle}\dot{M}_0 \text{ in } \dot{M}_1$ with $!^{\square}, !^{\triangle}$ dual, and $\dot{N} \equiv \dot{M}_1[x := \dot{M}_0]$. Aside from the syntax, this case is just as in the previous one: by the definition of a decreasing weighting plus Lemma A.7 we have $\|\dot{M}_1[x := \dot{M}_0]\| \leq \|\dot{M}_1\|$. Then similarly we have the result with

$$\begin{aligned} & \|^{[\beta!^i]}\text{let } !^i x = !^{\square}!^{\triangle}\dot{M}_0 \text{ in } \dot{M}_1\| \\ = \;& 2\|\dot{M}_0\| + \|\dot{M}_1\| \\ > \;& \|\dot{M}_1\| \\ \geq \;& \|\dot{M}_1[x := \dot{M}_0]\| \ . \end{aligned}$$

4. *Top level marked* $(\star!^{\square}\!\multimap)$ *contractions.* We consider the case of steps of rank 1; higher ranked steps can be seen as just multiple rank 1 steps.

$$\begin{aligned} & \|^{[\star!^{\square}\multimap_1]}(\text{let } !^{\square}x = \dot{L} \text{ in } \dot{M})\ \dot{N}\| \\ = \;& 4\|\dot{L}\| + 2\|\dot{M}\| + 2\|\dot{N}\| \\ > \;& 2\|\dot{L}\| + 2\|\dot{M}\| + 2\|\dot{N}\| \\ = \;& \|\text{let } !^{\square}x = \dot{L} \text{ in } (\dot{M}\ \dot{N})\| \ . \end{aligned}$$

5. *Top level marked* $(\star!^{\square}!^{\triangle})$ *contraction.* Again considering rank 1 steps, this case is straightforward:

$$\begin{aligned} & \|^{[\star!^{\square}!^{\triangle}{}_1]}\text{let } !^{\square}x = \text{let } !^{\triangle}y = \dot{L} \text{ in } \dot{M} \text{ in } \dot{N}\| \\ = \;& 4\|\dot{L}\| + 2\|\dot{M}\| + \|\dot{N}\| \\ > \;& 2\|\dot{L}\| + 2\|\dot{M}\| + \|\dot{N}\| \\ = \;& \|\text{let } !^{\triangle}y = \dot{L} \text{ in let } !^{\square}x = \dot{M} \text{ in } \dot{N}\| \ . \end{aligned}$$

6. *Top level marked* $(\star!^i!^{\square}!^w)$ *contraction.* We have a simple analysis just as in the case for $(\star!^{\square}!^{\triangle})$ steps:

$$\begin{aligned} & \|^{[\star!^i!^{\square}!^{w}{}_n]}\text{let } !^i x = !^{\square}(\text{let } !^i y = !^w \dot{L} \text{ in } \dot{M}) \text{ in } \dot{N}\| \\ = \;& 4\|\dot{L}\| + 2\|\dot{M}\| + \|\dot{N}\| \\ > \;& 2\|\dot{L}\| + 2\|\dot{M}\| + \|\dot{N}\| \\ = \;& \|\text{let } !^i y = !^w \dot{L} \text{ in let } !^i x = !^{\square}\dot{M} \text{ in } \dot{N}\| \ , \end{aligned}$$

again with n-rank steps just as n 1-rank steps.

For non-top level reductions, the inductive step is simple since the norm of a term increases directly with the norm of its subcomponents. Considering application, let $\dot{L}\ \dot{M} \to \dot{L}\ \dot{N}$ with $\|\dot{M}\| > \|\dot{N}\|$, and we have

$$\begin{aligned} & \|\dot{L}\ \dot{M}\| \\ = \;& 2\|\dot{L}\| + 2\|\dot{M}\| \\ > \;& 2\|\dot{L}\| + 2\|\dot{N}\| \\ = \;& \|\dot{L}\ \dot{N}\| \ . \end{aligned}$$

The other cases are equally simple. ♦

The following argument occurs twice in the subsequent lemma, so we present it just once right here.

Lemma A.9 *Let* $\dot{M}, \dot{N} \in \mathsf{SLin}^{\star\bullet}$ *both have decreasing weighting where for all occurrences of* x^i *in* $\dot{M}$ *we have* $i > \|\dot{N}\|$. *Then* $\dot{M}[x := \dot{N}]$ *has a decreasing weighting.*

Proof: Let $\dot{L}$ be a marked β-redex in $\dot{M}[x := \dot{N}]$, so either $\dot{L} \equiv {}^{[\beta\multimap]}(\lambda y.\, \dot{L}_0)\ \dot{L}_1$, $\dot{L} \equiv {}^{[\beta!^{\square}]}\text{let } !^{\square}y = !^{\square}\dot{L}_1 \text{ in } \dot{L}_0$ or $\dot{L} \equiv {}^{[\beta!^i]}\text{let } !^i y = !^{\square}!^{\triangle}\dot{L}_1 \text{ in } \dot{L}_0$ for $!^{\square}, !^{\triangle}$ dual in the latter case. If the considered occurrence of $\dot{L}$ comes from $\dot{M}$, the result follows from Lemma A.7: since $\|\dot{L}_1[x := \dot{N}]\| < \|\dot{L}_1\|$ by the hypothesis, the condition for decreasing weighting involving y continues to hold. If the considered occurrence of $\dot{L}$ comes from $\dot{N}$, then the result is trivial since it is unchanged. ♦

Lemma A.10 *Let* $\dot{M} \xrightarrow[\mathsf{SLin}^{\star\bullet}_0]{} \dot{N}$ *be a top-level contraction, and let* $\dot{M}$ *have a decreasing weighting. Then* $\dot{N}$ *has a decreasing weighting.*

Proof: We consider each of the four possible reduction rules separately. In each case, we show that every marked $(\beta\multimap, \beta!)$ redex in $\dot{N}$ again satisfies the criteria for a decreasing weighting. This lemma and Lemma A.11 below have the same role and reasoning as Barendregt's proof of the similar result for call-by-name [5, Lemma 11.2.18.(ii)]. Since our system is somewhat more complicated, we decompose the result into two lemmas, first for marked redexes within the body of the contractum, and then for redexes enclosing it.

1. *Marked top-level* $(\beta\multimap)$ *reduction.* So $\dot{M} \equiv {}^{[\beta\multimap]}(\lambda y.\, \dot{M}_0)\ \dot{M}_1$, and $\dot{N} \equiv \dot{M}_0[y := \dot{M}_1]$ has a decreasing weighting by Lemma A.9.

2. *Marked top-level* $(\beta!^{\square})$ *reduction.* By the same reasoning as for marked top-level $(\beta\multimap)$ reduction.

3. *Marked top-level* $(\star!^{\square}\!\multimap)$ *reduction.* So $\dot{M} \equiv {}^{[!^{\square}\multimap]}(\dot{P}\ \dot{M}_0)\ \dot{M}_1$, and we again let $\dot{L}$ be a marked $(\beta\multimap, \beta!^{\square})$-redex in $\dot{N}$; again $\dot{L}$ is necessarily the residual of a marked redex $\dot{L}_0$ with decreasing weighting in $\dot{M}$. If $\dot{L}_0$ is a subterm of P, $\dot{M}_0$ or $\dot{M}_1$ then the result is trivial, since $\dot{L}$ is just the same as $\dot{L}_0$, just moved around within the term.

If on the other hand $\dot{L}_0 \equiv (P\,\dot{M}_0)$, then we must also have $\dot{L}_0 \equiv {}^{[\beta!^{\Box}]}\mathrm{let}\ !^{\Box}y = !^{\Box}\dot{L}_1$ in $\dot{M}_0$ and so $\dot{L} \equiv {}^{[\beta!^{\Box}]}\mathrm{let}\ !^{\Box}y = !^{\Box}\dot{L}_1$ in $(\dot{M}_0\,\dot{M}_1)$; since y does not appear free in $\dot{M}_1$ the decreasing weighting condition on $\dot{L}$ is again satisfied.

4. *Marked top-level* $(\star!^{\Box}!^{\triangle}, \star!^{i}!^{\Box}!^{w})$ *reduction.* By a similar analysis as for marked top-level $(\star!^{\Box}\!\multimap)$ reduction.

⧫

We can now state the general result that general marked reduction preserves the property of having a decreasing weighting.

Lemma A.11 *Let* $\dot{M} \xrightarrow[\mathsf{SLin}^{\star\bullet}_0]{} \dot{N}$*, where* $\dot{M}$ *has a decreasing weighting. Then* $\dot{N}$ *has a decreasing weighting.*

Proof: We proceed by structural induction on $\dot{M}$. The base case is a top-level redex, covered by Lemma A.10 above. The inductive steps correspond to the compatible closure of the redex within non-contracted contexts; these steps are non-trivial only for enclosure within marked $(\beta\!\multimap, \beta!^{\Box})$ redexes, which we consider presently.

1. *Marked top-level* $(\beta\!\multimap)$ *redexes.* For $\dot{M}$ we have ${}^{[\beta\multimap]}(\lambda x.\,\dot{M}_0)\dot{M}_1$, with two further cases depending on which subterm is contracted.
   (a) *Reduction* $\dot{M}_0 \to \dot{N}_0$. So we have $\dot{N} \equiv {}^{[\beta\multimap]}(\lambda x.\,\dot{N}_0)\,\dot{M}_1$. Let x^i occur in $\dot{N}_0$; then we must have x^i occurring in $\dot{M}_0$ as well since reduction will not create new weightings of a free variable, and by the initial conditions this $i > \|\dot{M}_1\|$.
   (b) *Reduction* $\dot{M}_1 \to \dot{N}_1$. In this case we have $\dot{N} \equiv {}^{[\beta\multimap]}(\lambda x.\,\dot{M}_0)\,\dot{N}_1$. Let x^i occur in $\dot{M}_0$; then by the initial decreasing weighting condition we have $i > \|\dot{M}_1\|$. Moreover by Lemma A.8 we have that $\|\dot{M}_1\| > \|\dot{N}_1\|$, and so $i > \|\dot{N}_1\|$ and the weighting is again decreasing.
2. *Marked top-level* $(\beta!^{\Box})$ *redexes.* By similar reasoning as for marked top-level $(\beta\!\multimap)$ redexes.

The cases of commuting conversion reduction rules is clear, and the lemma follows. ⧫

Lemma A.12 *Let* $M' \in \mathsf{SLin}^{\star\prime}$*. Then there exists a term* $\dot{M} \in \mathsf{SLin}^{\star\bullet}$*,* $|\dot{M}| \equiv M'$*, such that* $\dot{M}$ *has a decreasing weighting.*

Proof: Considering a term with n free variables all of weighting m, it is clear that the maximum norm of such a term occurs in a left-nested (or equivalently, right-nested) series of applications of one such variable (or abstractions and/or !-enclosures thereof) to another, *i.e.* $(\ldots((x_1^m\ x_2^m)\ x_3^m)\ x_{n-1}^m)\ x_n^m$, with norm $(2\cdot 2^{n-1} + 2^{n-2} + 2^{n-3} + 2)m$. So to give a variable weight greater than a term whose weight is at least the above quantity, $(\sum_{i=0}^{n} 2^i)m$ should suffice.

Correspondingly, to construct a decreasing weighting for a marked term M', we number all variable occurance in M' beginning with 1 from right to left, except in an eliminator where we number the bound term first. We then give a term numbered i the weight f_i, where f is the recurrance relation defined by the following two equations:

$$\begin{aligned} f_1 &= 1 \\ f_n &= \left(\sum_{i=0}^{n} 2^i\right)\cdot f_{n-1}\ , \qquad n > 1 \end{aligned}$$

By the above arguments this weighting is clearly decreasing. ⧫

Lemma A.13 $\mathsf{SLin}^{\star\prime}_0$ *reduction is strongly normalizing.*

Proof: By Lemma A.12 every term $M' \in \mathsf{SLin}^{\star\prime}$ has a decreasing weighting $\dot{M}$, and so by Lemma A.8 and Lemma A.11 all $\mathsf{SLin}^{\star\prime}_0$-sequences from M' are of length at most $\|\dot{M}\|$. ⧫

Corollary A.14 *All* $\mathsf{SLin}^{\star}$*-developments are finite.*

Proof: Immediate, since developments and $\mathsf{SLin}^{\star\prime}_0$ reduction are the same thing. ⧫

A.2 Unique Completions

Lemma A.15 $\mathsf{SLin}^{\star\prime}_0$ *reduction is weakly confluent.*

Proof: We take $M \xrightarrow[\mathsf{SLin}^{\star\prime}_0]{} M_0$, $M \xrightarrow[\mathsf{SLin}^{\star\prime}_0]{} M_1$ and construct the N such that

$$\begin{aligned} M_0 &\xrightarrow[\mathsf{SLin}^{\star\prime}_0]{}\!\!\!\!\!\rightarrow\ N. \text{ and} \\ M_1 &\xrightarrow[\mathsf{SLin}^{\star\prime}_0]{}\!\!\!\!\!\rightarrow\ N\ . \end{aligned}$$

We consider first the case of a top-level redex.

- *Top-level* $(\beta\!\multimap_0, \beta!^{\Box}_0)$ *steps.* For these top-level steps the result is immediate by Lemma A.4.
- *Top-level* $(\star!^{\Box}\!\multimap_0, \star!^{\Box}!^{\triangle}_0, \star!^{i}!^{\Box}!^{w}_0)$ *steps.* Each of these cases are simply an analysis of the various cases, which are difficult only in the bookkeeping. We present the details in the author's thesis [18], and omit the details here.

Cases with redexes in disjoint subterms and under compatible closure follow immediately. ⧫

Corollary A.16 $\mathsf{SLin}^{\star\prime}_0$ *reduction is confluent.*

Proof: Follows from Lemma A.13 and Corollary A.16. ⧫

Proposition A.17 *All* $\mathsf{SLin}^\star$*-developments are finite and can be extended to complete developments; all complete* $\mathsf{SLin}^\star$*-developments end in the same term.*

Proof: Finiteness and extension follow from FD(SLin); unique completion are by Lemma A.16 since development of markings and $\mathsf{SLin}^{\star\prime}_0$ reduction are the same, and since in a confluent system, normal forms are unique [5, Corollary 3.1.13]. ♦

Confluence

Although in the classical lambda calculi it is possible to deduce confluence directly from finite developments and unique completions, a subtle implicit assumption in those systems does not hold here. In (say) Name, given marked terms $(M, \bar{\gamma})$ and $(M, \bar{\delta})$, we will always have that $(M, \bar{\gamma} \cup \bar{\delta})$ is a valid marked term, and (trivally) that the original two terms occur as partial developments of the latter. This property does not hold for SLin: given two ranked marking of the same redex but different ranks, the simple set-theoretic union of the indexing sets does not produce a marked term. We say that two marking sets $\bar{\gamma}_1, \bar{\gamma}_2$ for a term M are *joinable* if there exists some set $\bar{\gamma}_0$ such that each of the first two occur as partial developments of the third, and that all markings are joinable if such a set exists for all markings of all terms. We discuss joinability in more detail and in other contexts in our thesis [18].

Lemma A.18 *All* $\mathsf{SLin}^\star$*-markings are joinable.*

Proof: It is clear that we can produce a valid marking by taking the higher-ranked of two ranked redexes of the same term, since the lesser ranked is a development of the greater ranked redex. ♦

Proposition A.19 *Reduction in* SLin *— excluding* $(\mathrm{W}!^i)$ *steps — is confluent.*

Proof: By standard techniques with Lemma A.17 and Lemma A.18. ♦

Lemma A.20 *Reduction of* SLin *terms by* $(\mathrm{W}!^i)$ *steps is confluent.*

Proof: Trivially, we can show by inspection that if $M \xrightarrow[(\mathrm{W}!^i)]{} M_1$ and $M \xrightarrow[(\mathrm{W}!^i)]{} M_2$, then we have some N such that for both i, $M_i \xrightarrow[(\mathrm{W}!^i)]{}{}^{=} N$. The full result follows by induction. ♦

Lemma A.21 *Let* $M, M_1, M_2 \in \mathsf{SLin}$*, with* $M \xrightarrow[(\mathrm{W}!^i)]{} M_1$ *and* $M \xrightarrow[\mathsf{SLin}]{} M_2$*, with the latter not by a* $(\mathrm{W}!^i)$ *step. Then there exists some* N *such that* $M_1 \xrightarrow[\mathsf{SLin}]{}{}^{=} N$ *and* $M_2 \xrightarrow[(\mathrm{W}!^i)]{}{}^{=} N$ *where the former is not by a* $(\mathrm{W}!^i)$ *step (where the* $^=$ *indicates reflexive closure).*

Proof: By an easy inspection of the different possible SLin steps and relative redex positions. ♦

Proof of Proposition 2.4: *Reduction in* SLin *is confluent.*
Follows from Lemma A.19, Lemma A.20 and Lemma A.21.

B Standard Evaluation

We have the usual, straightforward unique evaluation context lemma for SLin, although the larger language of terms means that this proof, as well as most of the proofs in this section, is notably more complicated than the standardization proofs of call-by-name, by-value or by-need.

Lemma B.1 *Let* $M \in \mathsf{SLin}$*. Then exactly one of the following is true:*

1. M *is an answer in* SLin*.*
2. M *demands some* $x \in \mathrm{fv}(M)$*.*
3. *There exists a unique top-level standard* SLin *redex* Δ*, plus some unique evaluation context* E *such that* $M \equiv E[\Delta]$*.*

Proof: By structural induction on M.

- $M \equiv x$. Clearly only Clause 2 is possible, with $E \equiv [\,]$.
- $M \equiv \lambda x.\, M_0$. All function are answers, so we have Clause 1 and clearly none other.
- $M \equiv M_0\, M_1$. Clause 1 does not apply; no application can be an answer. Applying the induction hypothesis on M_0, if Clause 1 applies for the subterm, then the top-level redex is a top-level $(\beta\multimap, !^i\multimap)$ standard step. If Clause 2 or 3 applies for the subterm, then the same applies for the whole term as well.
- $M \equiv !^w M_0$. Then clearly only Clause 1 applies.
- $M \equiv (\mathrm{let}\ !^w x = M_0\ \mathrm{in}\ M_1)$. Trivially, M is not an answer; we consider M_0 according to the induction hypothesis. If Clause 1 holds for M_0, then depending on the two possible forms for a well-typed subterm we have a top-level standard $(\beta!^w, !^w!^i)$ step. Otherwise, whichever clause holds for M_0 holds for M as well.
- $M \equiv !^c M_0$ *or* $M \equiv (\mathrm{let}\ !^c x = M_0\ \mathrm{in}\ M_1)$. As in the previous two cases.
- $M \equiv (\mathrm{let}\ !^i x = M_0\ \mathrm{in}\ M_1)$. This final case is rather complicated. We consider M_0 according to the induction hypothesis. If Clause 2 or 3 applies to M_0, then the same applies to M. Otherwise if M_0 is an answer, we have an analysis of its exact form:
  - $M_0 \equiv !^w N_0$. This first case is the most complicated. Considering M_1 inductively, if M_1 demands x, then we must further consider the clause which applies for N_0. If N_0 is

an answer, then M is a top-level standard $(\beta!^i, !^i!^w!^w)$ step, with the specific rule depending on the form of N_0. If N_0 demands a free variable or contains a standard redex, then the same applies to M.

Otherwise, if M_1 does not demand x, then whatever rule applied for M_1 applies for M as well.

– $M_0 \equiv !^c N_0$. If N_0 is an answer than we have M a top-level redex, and the specific rule is dictated by the form of N_0:
  * $N_0 \equiv !^w N_1$. Here, the $(\beta!^i)$ rule applies.
  * $N_0 \equiv (\text{let } !^i y = !^w N_1 \text{ in } A)$. In this case, we can apply the $(!^i!^c!^w)$ rule.

  Otherwise if Clause 2 or 3 applies to N_0, then the same applies to M.

– $M_0 \equiv (\text{let } !^i y = !^w N_0 \text{ in } A)$. We have that M is a top-level standard $(!^i!^i)$ step.

No other form of M is possible, and the proof is complete. ♦

Figure 32 shows the notion of reduction for $\mathsf{SLin}^\star$, derived from evaluation in SLin just by moving to the corresponding $\star$-steps of arbitrary rank.

Lemma B.2 *Let $M, N \in \mathsf{SLin}/\mathsf{SLin}^\star$. Then $M \xmapsto{\mathsf{SLin}}\!\!\!\!\rightarrow N$ if and only if $M \xmapsto{\mathsf{SLin}^\star}\!\!\!\!\rightarrow N$.*

Proof: Clear from the definitions of the rules; we simply exchange each n-rank $\star$ step in $\mathsf{Need}^\star$ for n ordinary steps in Need. ♦

Lemma B.3 *Let $M \in \mathsf{SLin}/\mathsf{SLin}^\star$. Then exactly one of the following is true:*

1. *M is an answer in $\mathsf{SLin}/\mathsf{SLin}^\star$.*
2. *M demands some $x \in \mathrm{fv}(M)$.*
3. *There exists at least one top-level standard $\mathsf{SLin}^\star$ redex Δ, plus some unique evaluation context E such that $M \equiv E[\Delta]$.*

Proof: As in Lemma B.1. ♦

Lemma B.4 *Let A be a $\mathsf{Need}/\mathsf{Need}^\star$ answer. Then $A[x := M]$ is also a $\mathsf{Need}/\mathsf{Need}^\star$ answer.*

Proof: Easy; free variables do not appear in the definition of answers, and so the result is an easy structural induction, observing that abstractions (respectively $!^w$-prefixed, $!^c$-prefixed terms) remain abstractions (respectively $!^w$-prefixed, $!^c$-prefixed terms). ♦

Lemma B.5 *Let $M, N \in \mathsf{Need}^\star$ where $M[x := N]$ is a $\mathsf{Need}/\mathsf{Need}^\star$ answer and* either

1. *N is not an answer, or*
2. *M does not demand x,*

or both. Then M is a $\mathsf{Need}/\mathsf{Need}^\star$ answer.

Proof: Clear from analysis of the particular M. ♦

Lemma B.6 *Let $M, N \in \mathsf{Need}/\mathsf{Need}^\star$, where M demands x. Then $M[y := N]$ demands x.*

Proof: Clear by induction of the evaluation context E such that $M \equiv E[x]$. ♦

Lemma B.7 *Let $M, N \in \mathsf{Need}^\star$ where $M[y := N]$ demands x and* either

1. *N does not demand x, or*
2. *M does not demand y,*

or both. Then M demands x.

Proof: By analysis of M and the E where $E[x] \equiv M[y := N]$. ♦

Lemma B.8 *Let $M \xmapsto[\mathsf{SLin}^\star]{} N$ be top-level. Then $M[x := L] \xmapsto[\mathsf{SLin}^\star]{} N[x := L]$ is also top-level and standard.*

Proof: Trivial, by inspection of the reduction rules. ♦

Lemma B.9 *Let $\sigma : M \xrightarrow[i]{} N$ in $\mathsf{SLin}^\star$ with $z \notin \mathrm{fv}(M, N)$. Then M demands z if and only if N demands z.*

Proof: We take $M \equiv E[z]$ and $\sigma : M \equiv C[\Delta] \xrightarrow[i]{} C[\Delta'] \equiv N$, and proceed with an analysis of C. If the top-level structure of C is that of a non-empty evaluation context (for example $C \equiv C_0\, M_0$), then we have the result by the induction hypothesis; if its top-level structure is not that of an evaluation context (for example $C \equiv M_0\, C_0$), then the result is immediate. Otherwise, for top-level σ we consider the specific rule which applies.

1. $M \equiv (\text{let } !^i x = !^w !^c M_0 \text{ in } M_1) \xrightarrow[i\text{-}(\beta !^i_{wc})]{} M_1[x := M_0] \equiv N$, *where M_1 does not demand x.* We must have M_1 demanding z, since no subterm of M_0 could be in evaluation position; the result then follows from Lemma B.6 and Lemma B.7.
2. $M \equiv (\text{let } !^i x = !^w M_0 \text{ in } M_1) \xrightarrow[i\text{-}(\mathrm{W}!^i)]{} M_1 \equiv N$ *where $x \notin \mathrm{fv}(M_1)$.* Trivially, if $x \notin \mathrm{fv}(M_1)$ then it must be that M_1 demands z.
3. $M \equiv ({}^{\mathrm{i}^w}\mathcal{P}^n\, M_1)\, M_2 \xrightarrow[i\text{-}(\star !^i \multimap)]{} {}^{\mathrm{i}^w}\mathcal{P}^n\, (M_1\, M_2) \equiv N$ *where M_1 is not an answer.* We have two possibilities. If M_1 demands z, then the result is immediate. Otherwise, taking

$$ {}^{\mathrm{i}^w}\mathcal{P}_j \equiv \text{let } !^i x_j = !^w L_j \text{ in } ,$$

we have a strictly decreasing sequence a_k, $1 \leq k \leq m$, $1 \leq a_k \leq n$, such that M_1 demands $x_{(a_1)}$, $L_{(a_j)}$ demands $x_{(a_{j+1})}$ and $L_{(m)}$ demands z. This arrangement would clearly be preserved from one side of the reduction arrow to the other.

Syntactic domains As in SLin.

Reduction rules

$(\beta\multimap)$	$(\lambda x.\, M)\, N$	$\xmapsto[\mathsf{SLin}]{}$	$M[x := N]$
$(\beta!^c)$	let $!^c x = !^c N$ in M	$\xmapsto[\mathsf{SLin}]{}$	$M[x := N]$
$(\beta!^w)$	let $!^w x = !^w N$ in M	$\xmapsto[\mathsf{SLin}]{}$	$M[x := N]$
$(\beta!^i_{cw})$	let $!^i x = !^c!^w N$ in M	$\xmapsto[\mathsf{SLin}]{}$	$M[x := N]$
$(\beta!^i_{wc})$	let $!^i x = !^w!^c N$ in $E[x]$	$\xmapsto[\mathsf{SLin}]{}$	$(E[x])[x := N]$
$(\star!^i\multimap)$	$({}^{\mathrm{i}^w}\mathcal{P}^n\, A)\, N$	$\xmapsto[\mathsf{SLin}]{}$	${}^{\mathrm{i}^w}\mathcal{P}^n\, (A\, N)$
$(\star!^\Box!^i)$	let $!^\Box x = ({}^{\mathrm{i}^w}\mathcal{P}^n\, A)$ in M where $!^\Box$ ranges over $!^w$, $!^c$, $!^i$.	$\xmapsto[\mathsf{SLin}]{}$	${}^{\mathrm{i}^w}\mathcal{P}^n\, (\text{let } !^\Box x = A \text{ in } M)$
$(!^i!^w!^w)$	let $!^i x = !^w({}^{\mathrm{i}^w}\mathcal{P}\, A)$ in $E[x]$	$\xmapsto[\mathsf{SLin}]{}$	${}^{\mathrm{i}^w}\mathcal{P}\, (\text{let } !^i x = !^w A \text{ in } E[x])$
$(!^i!^c!^w)$	let $!^i x = !^c({}^{\mathrm{i}^w}\mathcal{P}\, A)$ in N	$\xmapsto[\mathsf{SLin}]{}$	${}^{\mathrm{i}^w}\mathcal{P}\, (\text{let } !^i x = !^c A \text{ in } N)$

Figure 32: Evaluation in $\mathsf{SLin}^\star$.

4. $M \equiv (\mathcal{R}^n\, {}^{\mathrm{i}^w}\mathcal{P}\, M_1)\, M_2 \xrightarrow[i\text{-}(\star!^i\multimap)]{} \mathcal{R}^n\, {}^{\mathrm{i}^w}\mathcal{P}\, (M_1\, M_2) \equiv N$ *where not all of the* $\mathcal{R}$ *are of the form* ${}^{\mathrm{i}^w}\mathcal{P}$. This case is similar to the previous one. We must have some $0 \le m < n$ such that the $\mathcal{R}^n$ can be partitioned as

$$\mathcal{R}^n \equiv {}^{\mathrm{i}^w}\mathcal{R}^m\, \mathcal{R}_{m+1}\, \mathcal{R}^{n-m-1} \ ,$$

where $\mathcal{R}_{m+1}$ is *not* of the form ${}^{\mathrm{i}^w}\mathcal{R}_{m+1}$. Hence, its bound term L is in evaluation position. Then either L demands z, or we have a series a_k among the bound variables and terms of ${}^{\mathrm{i}^w}\mathcal{R}^m$ as in the previous case which end in a bound term demanding z.

5. $M \equiv (\mathcal{P}^n \text{let } !^\Box x = M_0 \text{ in } M_1)\, M_2 \xrightarrow[i\text{-}(!^\Box\multimap)]{} \mathcal{P}^n\, (\text{let } !^\Box x = M_0 \text{ in } M_1\, M_2) \equiv N$ *where* $!^\Box$ *ranges over* $!^w, !^c$. As in Case 4.

6. $M \equiv (\text{let } !^\Box x = ({}^{\mathrm{i}^w}\mathcal{P}^n\, M_1) \text{ in } M_2) \xrightarrow[i\text{-}(\star!^\Box!^i)]{} {}^{\mathrm{i}^w}\mathcal{P}^n\, (\text{let } !^\Box x = M_1 \text{ in } M_2) \equiv N$, *where* M_1 *is not an answer.* As in Case 3, since M_1 must itself be demanded.

7. $M \equiv (\text{let } !^\Box x = (\mathcal{P}^n\, M_1) \text{ in } M_2) \xrightarrow[i\text{-}(\star!^\Box!^\triangle)]{} \mathcal{P}^n\, (\text{let } !^\Box x = M_1 \text{ in } M_2) \equiv N$, *where not all* $\mathcal{P}$ *are of the form* ${}^{\mathrm{i}^w}\mathcal{P}$. As in Case 4.

8. $M \equiv (\text{let } !^i x = !^c({}^{\mathrm{i}^w}\mathcal{P}^n\, M_1) \text{ in } M_2) \xrightarrow[i\text{-}(\star!^i!^c!^w)]{} {}^{\mathrm{i}^w}\mathcal{P}^n\, (\text{let } !^i x = !^c M_1 \text{ in } M_2) \equiv N$. This case is reminiscent of Case 6. We have either M_2 demanding z, or else some chain of demand as in Case 3.

9. $M \equiv (\text{let } !^i x = !^w({}^{\mathrm{i}^w}\mathcal{P}^n\, M_2) \text{ in } M_3) \xrightarrow[i\text{-}(\star!^i!^w!^w)]{} {}^{\mathrm{i}^w}\mathcal{P}^n\, (\text{let } !^i x = !^w M_2 \text{ in } M_3) \equiv N$, *where* M_2 *is not an answer.* We have either M_2 demanding z, or else M_2 demanding x with one of the arrangements of the previous case.

10. $M \equiv (\text{let } !^i x = !^w({}^{\mathrm{i}^w}\mathcal{P}^n\, A) \text{ in } M) \xrightarrow[i\text{-}(\star!^i!^w!^w)]{} {}^{\mathrm{i}^w}\mathcal{P}^n\, (\text{let } !^i x = !^w A \text{ in } M) \equiv N$ *where* M *does not demand* x. As before, except that the only possible arrangement is that M demands z.

Finally, it is not possible for a top-level $(\beta\multimap, \beta!^c, \beta!^w, \beta!^i_{cw})$ step to be internal, so the proof is complete. ♦

Lemma B.10 *Let* $\sigma : M \xrightarrow[i]{} N$ *in* $\mathsf{SLin}^\star$. *Then* M *is an answer if and only if* N *is an answer.*

This lemma motivates the restriction to the (!!!) rules to which we alluded on page 10 but did not completely explain. We rejected rules of the form

$$\text{let } !^i x = !^\Box(\text{let } !^i y = !^c M_1 \text{ in } M_2) \text{ in } M_3 \xrightarrow[i\text{-}(!^i!^\Box!^c)]{} \text{let } !^i y = !^c M_1 \text{ in } (\text{let } !^i x = !^\Box M_2 \text{ in } M_3) \ ,$$

where $!^\Box$ ranges over $!^w, !^c$. Taking the case where $!^\Box$ is $!^w$, if M_3 is an answer then this rule will take an answer to a non-answer, which should not happen under any circumstances. Rather than convoluting the definitions of standard reduction and answer to step around this problem, we simply disallow these rules, which are not required for any other properties of the calculus or translations.

Proof: By structural induction on M.

- $M \equiv (\text{let } !^i x = !^w!^c M_0 \text{ in } M_1) \xrightarrow[i\text{-}(\beta!^i_{wc})]{} M_1[x := M_0] \equiv N$, *where* M_1 *does not demand* x. M is an answer if and only if M_1 is an answer. Since M_1 does not demand x, by Lemmas B.4 and B.5 we have that M_1 is an answer if and only if $M_1[x := M_0] \equiv N$ is an answer.
- $M \equiv (\text{let } !^i x = !^w M_0 \text{ in } M_1) \xrightarrow[i\text{-}(\mathrm{W}!^i)]{} M_1 \equiv N$ *where* $x \notin \mathrm{fv}(M_1)$. Clearly both sides are answers exactly when M_1 is an answer.

- $M \equiv (\mathcal{P}^n\, M_1)\; M_2 \xrightarrow[i\text{-}(\star!^{\square}\text{—}\circ)]{} \mathcal{P}^n\,(M_1\; M_2) \equiv N$ *where M_1 is not an answer.* Trivially, neither side is an answer.
- $M \equiv (\text{let } !^{\square}x = (\mathcal{P}^n\, M_1) \text{ in } M_2) \xrightarrow[i\text{-}(\star!^{\square}!^{\triangle})]{} \mathcal{P}^n\,(\text{let } !^{\square}x = M_1 \text{ in } M_2) \equiv N$, *where M_1 is not an answer.* Again, neither side is an answer, since even if $!^{\square}$ were $!^i$, we would still not have its bound term on either side under a $!^w$ prefix.
- $M \equiv (\text{let } !^i x = !^{\square}(\overset{w}{\mathrm{i}}\mathcal{P}^n\, M_1) \text{ in } M_2) \xrightarrow[i\text{-}(\star!^i!^{\square}!^w)]{} \overset{w}{\mathrm{i}}\mathcal{P}^n\,(\text{let } !^i x = !^{\square}M_1 \text{ in } M_2) \equiv N$ *where $!^{\square}$ ranges over $!^w, !^c$ and either M_1 is not an answer, M_2 does not demand x, or both.* If $!^{\square}$ is $!^c$ then the neither side is an answer. Otherwise both sides are clearly answers exactly when M_2 is an answer.

We have compatible closure by the induction hypothesis and definition of answer, and so the proof is complete. ♦

Lemma B.11 *Let γ, δ mark distinct internal $\mathsf{SLin}^\star$ redexes in M, $\sigma : M \xrightarrow[(\gamma)]{} N$. Then every member of δ/σ is internal.*

Proof: Again, by induction on the structure of M. If $\gamma \equiv \delta$ then there are no residuals, and the result is vacuously true, so we assume from now on that the two are different in some way, *i.e.* in rank if not position. We begin with the case where γ is top-level, excluding as usual those rules of which top-level steps cannot be internal, and excluding as well ($\mathrm{W}!^i$) steps which cannot be marked.

- $M \equiv (\text{let } !^i x = !^w!^c M_0 \text{ in } M_1) \xrightarrow[i\text{-}(\beta!^i_{wc})]{} M_1[x := M_0] \equiv N$, *where M_1 does not demand x.* If δ indexes a redex in M_1, then the residual is just δ again, which clearly remains internal: if it is within a non-evaluation context at first, it will continue to be so under the substitution of a non-demanded variable. For δ indexing a term is M_0 the result is again clear; if M_1 does not demand x, then it does not demand the replacement for x.
- $M \equiv (\overset{w}{\mathrm{i}}\mathcal{P}^n\, M_1)\; M_2 \xrightarrow[i\text{-}(\star!^i\text{—}\circ)]{} \overset{w}{\mathrm{i}}\mathcal{P}^n\,(M_1\; M_2) \equiv N$ *where M_1 is not an answer.* A variety of cases arise; at the risk of being pedantic we catalog the possibilities by considering the explicit structure of the path δ.
  - $\delta \equiv \epsilon$. So δ is a $(\star!^{\square}\text{—}\circ)$ step of different rank m than γ but is also top-level. We compare m and n: if $m < n$, then δ/σ is empty. If $m > n$, then the single residual is the $(\star!^{\square}\text{—}\circ)$ in context $(\overset{w}{\mathrm{i}}\mathcal{P}^n\,[])$ with rank $(m-n)$. This step is clearly internal if and only if γ is internal as well.
  - $\delta \succcurlyeq \underline{@1}$. We distinguish three cases, respectively, reduction of a bound term, top-level reduction of a subterm $(\overset{w}{\mathrm{i}}\mathcal{P}_m \cdots \overset{w}{\mathrm{i}}\mathcal{P}_n\, M_1)$, and reduction within M_1.
    - $\delta \equiv \underline{@1}(\underline{\ell 2}^m)\underline{\ell 1}\underline{!1}\delta_0$, $m < n$, *i.e.* reduction of the bound term L in $\overset{w}{\mathrm{i}}\mathcal{P}_{m+1} \equiv (\text{let } !^i x = !^w L \text{ in})$. Since this redex is internal in M, either δ_0 is internal to L or if M_1 does not demand x. Whichever condition is true in M would continue to be true in N, so the residual will be internal as well.
    - $\delta \equiv \underline{@1}(\underline{\ell 2}^m)$, $m < n$. We have two possibilities, either a $(\star!^i!^w!^w)$ redex or a $(\beta!^i_{wc})$ redex. Taking $L_2 \equiv \overset{w}{\mathrm{i}}\mathcal{P}_{m+1} \cdots \overset{w}{\mathrm{i}}\mathcal{P}_n\, M_1$, in the first case we have
$$\begin{aligned}&(\overset{w}{\mathrm{i}}\mathcal{P}_m \cdots \overset{w}{\mathrm{i}}\mathcal{P}_n\, M_1)\\ \equiv\;& \text{let } !^i x_m = !^w(\overset{w}{\mathrm{i}}\mathcal{R}\, L_1) \text{ in } L_2\ .\end{aligned}$$
      Since this redex is internal, we must have either that L_2 does not demand x, or that L_1 is an answer, or both. L_2 demands x if and only if $\overset{w}{\mathrm{i}}\mathcal{P}_{m+1} \cdots \overset{w}{\mathrm{i}}\mathcal{P}_n\,(M_1\; M_2)$ demands x, so the residual is internal as well. For a $(\beta!^i_{wc})$ step, we have
$$\begin{aligned}&(\overset{w}{\mathrm{i}}\mathcal{P}_m \cdots \overset{w}{\mathrm{i}}\mathcal{P}_n\, M_1)\\ \equiv\;& \text{let } !^i x_m = !^w!^c L_1 \text{ in } L_2\ ,\end{aligned}$$
      which if internal means L_2 does not demand x, so again neither does $\overset{w}{\mathrm{i}}\mathcal{P}_{m+1} \cdots \overset{w}{\mathrm{i}}\mathcal{P}_n\,(M_1\; M_2)$, and so the residual is internal.
    - $\delta \equiv \underline{@1}(\underline{\ell 2}^n)\delta_0$. Since $((\overset{w}{\mathrm{i}}\mathcal{P}^n\,[])\; M_2)$ is an evaluation context, δ_0 must be internal to M_1, and so the residual must be internal as well.
  - $\delta \succcurlyeq \underline{@2}$. Contraction within M_2 in both M and N is clearly internal.
- $M \equiv (\mathcal{R}^n\, \overset{w}{\mathrm{i}}\mathcal{P}\, M_1)\; M_2 \xrightarrow[i\text{-}(\star!^i\text{—}\circ)]{} \mathcal{R}^n\, \overset{w}{\mathrm{i}}\mathcal{P}\,(M_1\; M_2) \equiv N$ *where not all of the $\mathcal{R}$ are of the form $\overset{w}{\mathrm{i}}\mathcal{P}$.*
- $M \equiv (\mathcal{P}^n \text{let } !^{\square}x = M_0 \text{ in } M_1)\; M_2 \xrightarrow[i\text{-}(\star!^{\square}\text{—}\circ)]{} \mathcal{P}^n\,(\text{let } !^{\square}x = M_0 \text{ in } M_1\; M_2) \equiv N$ *where $!^{\square}$ ranges over $!^w, !^c$.*

  These two cases rely on exactly the same reasoning as in the first ($!^i$—$\circ$) case. We must only distinguish the different $!^{\square}$ eliminator prefixes since their evaluation behavior, and thus the internal redexes which they admit, differ from a $!^i$-eliminator with an $!^w$-prefixed argument. It is clear though that the criteria for these redexes to be internal is preserved in both M and N.
- $M \equiv (\text{let } !^{\square}x = (\overset{w}{\mathrm{i}}\mathcal{P}^n\, M_1) \text{ in } M_2) \xrightarrow[i\text{-}(\star!^{\square}!^i)]{} \overset{w}{\mathrm{i}}\mathcal{P}^n\,(\text{let } !^{\square}x = M_1 \text{ in } M_2) \equiv N$, *where M_1 is not an answer.* We again analyze the structure of δ.
  - $\delta \equiv \epsilon$. As in the ($!^{\square}$—$\circ$) cases. A $(!^{\square}!^i)$ redex of lesser rank would have no residuals, while a $(!^{\square}!^{\triangle})$ redex of greater rank would clearly

remain internal: since both δ and γ are internal in M, either M_1 is not an answer, M_2 does not demand x, or both. Each condition would hold in N as in M, and hence the residual of δ would be internal.

- $\delta \succcurlyeq \underline{\ell_1}$.
  - $\delta \succcurlyeq \underline{\ell_1}(\underline{\ell_2}^m)\underline{\ell_1}$, $m < n$.
  - $\delta \equiv \underline{\ell_1}(\underline{\ell_2}^m)$, $m < n$. Both of these cases are as in the corresponding case for a $(!^{\square}\text{–}\circ)$ step.
  - $\delta \equiv \underline{\ell_1}(\underline{\ell_2}^m)\delta_0$. So δ_0 indexes an internal redex of M_1, which clearly makes the residual $(\underline{\ell_2}^n)\underline{\ell_1}\delta_0$ of δ again internal.
- $\delta \succcurlyeq \underline{\ell_2}$. Since M_1 is not an answer, contractions of M_2 in N are clearly internal.

- $M \equiv (\text{let } !^{\square}x = (\mathcal{P}^n\, M_1) \text{ in } M_2) \xrightarrow[i\text{-}(\star!^{\square}!^{\triangle})]{} \mathcal{P}^n\,(\text{let } !^{\square}x = M_1 \text{ in } M_2) \equiv N$, *where not all* $\mathcal{P}$ *are of the form* $\overset{w}{\mathrm{i}}\mathcal{P}$. This case is again like the first case of internal $(\star!^{\square}!^{i})$ reduction, noting variations in the criteria for internal contraction, which again γ preserves.
- $M \equiv (\text{let } !^{i}x = !^{\square}(\text{let } !^{i}y = !^{w}M_1 \text{ in } M_2) \text{ in } M_3) \xrightarrow[i\text{-}(!^{i}!^{\square}!^{w})]{} (\text{let } !^{i}y = !^{w}M_1 \text{ in } (\text{let } !^{i}x = !^{\square}M_2 \text{ in } M_3)) \equiv N$ *where* $!^{\square}$ *ranges over* $!^{w}, !^{c}$.
  - $\delta \equiv \epsilon$. As for $(\star!^{\square}!^{\triangle})$ steps of the same location but different rank.
  - $\delta \succcurlyeq \underline{\ell_1}$. Since $!^{\square}$-prefixed expressions cannot be top-level redexes, we in fact have $\delta \succcurlyeq \underline{\ell_1}\underline{!_1}$.
    - $\delta \equiv \underline{\ell_1}\underline{!_1}$. As in the $(\star!^{i}\text{–}\circ)$ case, we have either a $(\beta!^{i}_{wc})$ or $(\star!^{i}!^{w}!^{w})$ step, which is clearly again internal after contraction of γ.
    - $\delta \equiv \underline{\ell_1}\underline{!_1}$. This step is internal either if δ_1 is itself internal in the bound term, or if the bound variable is not demanded, or both. The two conditions are clearly both preserved by contracting γ, so the residual is again internal.
    - $\delta \succcurlyeq \underline{\ell_1}\underline{!_1}$. If M_2 demands x, then δ_1 must index an internal redex in M_1; otherwise if M_2 does not demand x the residual is trivially internal.
  - $\delta \succcurlyeq \underline{\ell_2}$. Trivially, δ_1 lust be internal in M_2, and so the residual of δ will be internal as well in N.

It is also possible that δ will be top-level when σ is not. In that case we have the result by a similar inspection of the individual cases; by the preceding Lemmas B.9 and B.10 on the effect of internal contractions on demand and answerhood, it is clear that the forms of subterms required for δ to be internal are upheld by contraction of the redex at γ. For δ, γ in distinct subterms it again follows from Lemmas B.9 and B.10 that if δ had been in a non-evaluation context, contracting γ would not cause the context to become an evaluation context. Finally, for δ, γ in the same subterm the result is immediate by the induction hypothesis. ♦

Lemma B.12 *Let* $\sigma : M_0 \xrightarrow[i]{} N_0$ *and let* $N_0 \mapsto N_1$, *both in* $\mathsf{SLin}^{\star}$. *Then* M_0 *has a standard redex: there exists some* M_1 *such that* $M_0 \mapsto M_1$.

Proof: So we have:

$$\begin{aligned} \sigma : M_0 \equiv C_0[\Delta_0] &\xrightarrow[i]{} C_0[\Delta_0'] \equiv N_0 \\ N_0 \equiv E_1[\Delta_1] &\mapsto E_1[\Delta_1'] \equiv N_1 \ , \end{aligned}$$

and we seek to construct or otherwise show the existence of the E, Δ such that $M_0 \equiv E[\Delta]$ and Δ is a standard step. We proceed by structural induction on M_0, and perform an analysis on C_0 and E_1.

We begin as usual where $C_0 \equiv [\,]$, excluding as usual the steps which cannot be both top-level and internal.

- *Where*

$$\begin{aligned} & M_0 \\ \equiv\ & (\text{let } !^{i}x = !^{w}!^{c}M_1 \text{ in } M_2) \\ \xrightarrow[i\text{-}(\beta!^{i}_{wc})]{}\ & M_2[x := M_1] \\ \equiv\ & N_0 \ , \end{aligned}$$

  with M_2 *does not demand* x. Since σ is internal, we must have that M_2 does not demand x. By Lemma B.5, since $M_2[x := M_1]$ has a standard redex, we know that M_2 is not an answer, and similarly by Lemma B.7 it could not demand any other variable, so by Lemma B.3 M_2 must have a standard redex, which would correspond to the contraction in N_0.
- $M_0 \equiv (\text{let } !^{i}x = !^{w}M_1 \text{ in } M_2) \xrightarrow[i\text{-}(\mathrm{W}!^{i})]{} M_2 \equiv N_0$ *where* $x \notin \mathrm{fv}(M_2)$. Trivially, $\Delta \equiv \Delta_1$ and $E \equiv \text{let } !^{i}x = !^{w}M_1 \text{ in } E_1$.
- $M_0 \equiv (\overset{w}{\mathrm{i}}\mathcal{P}^n\, M_1)\, M_2 \xrightarrow[i\text{-}(\star!^{i}\text{–}\circ)]{} \overset{w}{\mathrm{i}}\mathcal{P}^n\,(M_1\ M_2) \equiv N_0$ *where* M_1 *is not an answer.* $\Delta \equiv \Delta_1$, and E depends on E_1:
  - $E_1 \equiv \overset{w}{\mathrm{i}}\mathcal{P}^n\,(E_2\ M_2)$. Then $E \equiv (\overset{w}{\mathrm{i}}\mathcal{P}^n\, E_2)\, M_2$.
  - $E_1 \equiv \overset{w}{\mathrm{i}}\mathcal{P}^{m-1}\,(\text{let } !^{i}x_m = !^{w}E_2 \text{ in } \overset{w}{\mathrm{i}}\mathcal{P}^n_{m+1}\,(M_1\ M_2))$, where $\overset{w}{\mathrm{i}}\mathcal{P}^n_{m+1}\,(M_1\ M_2)$ demands x_m. Then $E \equiv (\overset{w}{\mathrm{i}}\mathcal{P}^{m-1}\,\text{let } !^{i}x_m = !^{w}E_2 \text{ in } \overset{w}{\mathrm{i}}\mathcal{P}^n_{m+1}\,M_1)\, M_2$.
- $M_0 \equiv (\mathcal{R}^n\,\overset{w}{\mathrm{i}}\mathcal{P}\,M_1)\, M_2 \xrightarrow[i\text{-}(\star!^{i}\text{–}\circ)]{} \mathcal{R}^n\,\overset{w}{\mathrm{i}}\mathcal{P}\,(M_1\ M_2) \equiv N_0$ *where not all of the* $\mathcal{R}$ *are of the form* $\overset{w}{\mathrm{i}}\mathcal{P}$, and
- *Where*

$$\begin{aligned} & M_0 \\ \equiv\ & (\mathcal{P}^n\,\text{let } !^{\square}x = M_0 \text{ in } M_1)\, M_2 \\ \xrightarrow[i\text{-}(!^{\square}\text{–}\circ)]{}\ & \mathcal{P}^n\,(\text{let } !^{\square}x = M_0 \text{ in } M_1\ M_2) \\ \equiv\ & N_0 \end{aligned}$$

with $!^{\square}$ *ranges over* $!^w, !^c$. Both of these cases are as in the first $(!^i \multimap)$ case, the difference being that for the $\mathcal{R}$ not of the form ${}^{\mathrm{i}w}\mathcal{P}$ the standard step would be in the bound term, rather than the free term, or else the β rule for a (let $!^w x = !^w M$ in), (let $!^c x = !^c M$ in) or (let $!^i x = !^c !^w M$ in) prefix.

- $M_0 \equiv (\text{let } !^{\square} x = ({}^{\mathrm{i}w}\mathcal{P}^n\, M_1) \text{ in } M_2) \xrightarrow[i\text{-}(\star !^{\square} !^i)]{} {}^{\mathrm{i}w}\mathcal{P}^n\,(\text{let } !^{\square} x = M_1 \text{ in } M_2) \equiv N_0$, *where* M_1 *is not an answer.* Since M_1 is not an answer, either the standard redex is within it, or is within a bound term in the ${}^{\mathrm{i}w}\mathcal{P}^n$. In both cases we have $\Delta \equiv \Delta_1$. In the former case we have simply $E_1 \equiv {}^{\mathrm{i}w}\mathcal{P}^n\,(\text{let } !^{\square} x = E_2 \text{ in } M_2)$, and so $E \equiv \text{let } !^{\square} x = ({}^{\mathrm{i}w}\mathcal{P}^n\, E_2) \text{ in } M_2$. In the latter case we have

$$N_0 \equiv {}^{\mathrm{i}w}\mathcal{P}^m\, \text{let } !^i x_m = !^w E_2[\Delta_1] \text{ in } {}^{\mathrm{i}w}\mathcal{P}^n_{m+1}\,(\text{let } !^{\square} x = M_1 \text{ in } M_2)$$

where $({}^{\mathrm{i}w}\mathcal{P}^n_{m+1}\,(\text{let } !^{\square} x = M_1 \text{ in } M_2))$ demands x_m. Then we can construct

$$E \equiv \text{let } !^{\square} x = {}^{\mathrm{i}w}\mathcal{P}^m\,(\text{let } !^i x_m = !^w E_2[\Delta_1] \text{ in } ({}^{\mathrm{i}w}\mathcal{P}^n_{m+1}\, M_1)) \text{ in } M_2 \ .$$

- $M_0 \equiv (\text{let } !^{\square} x = (\mathcal{P}^n\, M_1) \text{ in } M_2) \xrightarrow[i\text{-}(\star !^{\square} !^i)]{} \mathcal{P}^n\,(\text{let } !^{\square} x = M_1 \text{ in } M_2) \equiv N_0$, *where not all* $\mathcal{P}$ *are of the form* ${}^{\mathrm{i}w}\mathcal{P}$. Similar to the previous case, again with the caveat that non-${}^{\mathrm{i}w}\mathcal{P}$ prefixes have a different evaluation behavior.

- $M_0 \equiv (\text{let } !^i x = !^{\square}({}^{\mathrm{i}w}\mathcal{P}^n\, M_1) \text{ in } M_2) \xrightarrow[i\text{-}(!^i !^{\square} !^w)]{} ({}^{\mathrm{i}w}\mathcal{P}^n\,(\text{let } !^i x = !^{\square} M_1 \text{ in } M_2)) \equiv N_0$ *where* M_1 *is not an answer and* $!^{\square}$ *ranges over* $!^w, !^c$. If $!^{\square}$ is $!^c$, then either the redex is within M_1, or is by demand within a bound term of the ${}^{\mathrm{i}w}\mathcal{P}$. In the latter case we have an analysis an in the previous similar cases of this proof. If $!^{\square}$ is $!^i$, then we either have the standard step within M_2, or M_2 demanding x and one of the arrangements for reduction under the $!^w$.

- $M_0 \equiv (\text{let } !^i x = !^w({}^{\mathrm{i}w}\mathcal{P}^n\, A) \text{ in } M_1) \xrightarrow[i\text{-}(!^i !^w !^w)]{} ({}^{\mathrm{i}w}\mathcal{P}^n\,(\text{let } !^i x = !^w A \text{ in } M_1)) \equiv N_0$ *where* M_1 *does not demand* x. Trivially, the standard redex in each case in within M_1.

We can have no other top-level internal steps.

If $E_1 \equiv [\,]$, then the result is immediate, with $E \equiv [\,]$ as well.

Otherwise the result is clear. If the topmost configuration of C_0 is that of an evaluation context — that is, if C_0 and E_0 both have the same immediate structure such as

$$\begin{array}{rcl} C_0 & \equiv & C_1\, L \text{ and} \\ E_1 & \equiv & E_1\, L \ , \end{array}$$

then we have the result by the induction hypothesis. Finally, when the internal and standard contractions are in different subterms, we have the result immediately by Lemmas B.9 and B.10. ♦

Lemma B.13 *Let* $\sigma : M_0 \xrightarrow[i]{} N_0$ *and* $M_0 \xmapsto{\delta} N_1$ *in* $\mathsf{SLin}^\star$. *Then* δ/σ *contains a single element* δ_1 *which is a standard redex of* N_0*:* $N_0 \xmapsto{\delta_1} N$.

Proof: The analysis is similar to that of the previous proof, by a structural induction on M. Where both contractions are in the same subterm we have result immediately from the induction hypothesis. We have three primary cases which remain: where one or the other step is top-level, and where each is in a distinct child of the top-level term. We take

$$\begin{array}{rcl} \sigma : M_0 \equiv & C_0[\Delta_0] \xrightarrow[i]{} C_0[\Delta_0'] & \equiv N_0 \\ M_0 \equiv & E_1[\Delta_1] \mapsto E_1[\Delta_1'] & \equiv N_1 \ , \end{array}$$

and construct in each case the required E and Δ.

If the standard step is top-level, then the result is immediate from a consideration of the structure of the terms following from each possible redex, with Lemmas B.9 and B.10.

If the internal step is top-level, we have the following cases.

- *With*

$$\begin{array}{rcl} & M_0 & \\ \equiv & (\text{let } !^i x = !^w !^c M_1 \text{ in } M_2) & \\ \xrightarrow[i\text{-}(\beta !^i_{wc})]{} & M_2[x := M_1] & \\ \equiv & N_0 \ , & \end{array}$$

where M_1 *does not demand* x. Since M_1 does not demand x, the standard step must be within M_1, so we have the result by Lemmas B.6 and B.8.

- $M_0 \equiv (\text{let } !^i x = !^w M_1 \text{ in } M_2) \xrightarrow[i\text{-}(\mathrm{W} !^i)]{} M_2 \equiv N_0$ *where* $x \notin \mathrm{fv}(M_2)$. We can have only $E_1 \equiv (\text{let } !^i x = !^w M_1 \text{ in } E_2)$, and then E is just E_2.

- $M_0 \equiv ({}^{\mathrm{i}w}\mathcal{P}^n\, M_1)\, M_2 \xrightarrow[i\text{-}(\star !^i \multimap)]{} {}^{\mathrm{i}w}\mathcal{P}^n\,(M_1\, M_2) \equiv N_0$ *where* M_1 *is not an answer.* Either the standard step is within M_1 or a bound term of the ${}^{\mathrm{i}w}\mathcal{P}^n$. In the former case we have

$$E_1 \equiv ({}^{\mathrm{i}w}\mathcal{P}^n\, E_2)\, M_2 \ ,$$

and so $E \equiv {}^{\mathrm{i}w}\mathcal{P}^n\,(E_2\, M_2)$. In the latter case we have some $({}^{\mathrm{i}w}\mathcal{P}_{m+1} \cdots {}^{\mathrm{i}w}\mathcal{P}_n\, M_1)$ demanding x and

$$E_1 \equiv ({}^{\mathrm{i}w}\mathcal{P}^{m-1}\, \text{let } !^i x_m = !^w E_2 \text{ in } {}^{\mathrm{i}w}\mathcal{P}_{m+1} \cdots {}^{\mathrm{i}w}\mathcal{P}_n\, M_1)\, M_2$$

so then

$$E \equiv {}^{\mathrm{i}w}\mathcal{P}^{m-1}\,(\text{let } !^i x_m = !^w E_2 \text{ in } {}^{\mathrm{i}w}\mathcal{P}_{m+1} \cdots {}^{\mathrm{i}w}\mathcal{P}_n\,(M_1\, M_2)) \ .$$

- $M_0 \equiv (\mathcal{P}^n\, M_1)\; M_2 \xrightarrow[i\text{-}(\star!^{\Box}\multimap)]{} \mathcal{P}^n\,(M_1\; M_2) \equiv N_0$ *where not all of the* $\mathcal{P}$ *are of the form* $\overset{w}{\mathrm{i}}\mathcal{P}$. As in the previous case, taking into account the different evaluation behavior of the non $\overset{w}{\mathrm{i}}\mathcal{P}$ prefixes.
- $M_0 \equiv (\text{let } !^{\Box}x = (\overset{w}{\mathrm{i}}\mathcal{P}^n\, M_1) \text{ in } M_2) \xrightarrow[i\text{-}(\star!^{\Box}!^{i})]{} \overset{w}{\mathrm{i}}\mathcal{P}^n\,(\text{let } !^{\Box}x = M_1 \text{ in } M_2) \equiv N_0$, *where* M_1 *is not an answer,* and
- $M_0 \equiv (\text{let } !^{\Box}x = (\mathcal{P}^n\, M_1) \text{ in } M_2) \xrightarrow[i\text{-}(\star!^{\Box}!^{\triangle})]{} \mathcal{P}^n\,(\text{let } !^{\Box}x = M_1 \text{ in } M_2) \equiv N_0$, *where not all* $\mathcal{P}$ *are of the form* $\overset{w}{\mathrm{i}}\mathcal{P}$.

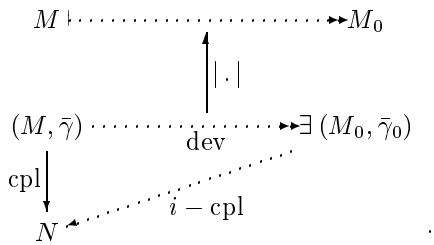
  Both of these cases are by a similar analysis as for the (!!) cases in the previous proof, with the obvious construction of the reshuffled evaluation contexts as in the $(!^{\Box}\multimap)$ steps immediately above.
- $M_0 \equiv (\text{let } !^{i}x = !^{\Box}(\overset{w}{\mathrm{i}}\mathcal{P}^n\, M_1) \text{ in } M_2) \xrightarrow[i\text{-}(!^{i}!^{\Box}!^{w})]{} \left(\overset{w}{\mathrm{i}}\mathcal{P}^n\,(\text{let } !^{i}x = !^{\Box}M_1 \text{ in } M_2)\right) \equiv N_0$ *where* M_1 *is not an answer and* $!^{\Box}$ *ranges over* $!^{w}, !^{c}$. If $!^{\Box}$ is $!^{c}$, then either the redex is within M_1, or is by demand within a bound term of the $\overset{w}{\mathrm{i}}\mathcal{P}$, and we have the same reshuffling of the evaluation contexts as in the $(!^{\Box}!^{\triangle})$ steps. Otherwise if $!^{\Box}$ is $!^{w}$ we have either the redex in $M_2 \equiv E_2[\Delta_1]$ and then $E \equiv \left(\overset{w}{\mathrm{i}}\mathcal{P}^n\,(\text{let } !^{i}x = !^{w}M_1 \text{ in } E_2)\right)$, or else M_2 demanding x and an arrangement as for when $!^{\Box}$ is $!^{c}$.
- $M_0 \equiv (\text{let } !^{i}x = !^{w}(\overset{w}{\mathrm{i}}\mathcal{P}^n\, A) \text{ in } M_1) \xrightarrow[i\text{-}(!^{i}!^{w}!^{w})]{} (\overset{w}{\mathrm{i}}\mathcal{P}^n\,(\text{let } !^{i}x = !^{w}A \text{ in } M_1)) \equiv N_0$ *where* M_1 *does not demand* x. Again, this final case is straightforward. We must have the standard step within $M_1 \equiv E_2[\Delta_1]$, and its residual in N_0 is just $(\overset{w}{\mathrm{i}}\mathcal{P}^n\,(\text{let } !^{i}x = !^{w}A \text{ in } E_2))$.

If the two redexes are in different subterms, then the standard step — being outside of the effect of the internal step — could not be duplicated or discarded, and so the residual is unique; by Lemmas B.9 and B.10 it is again a standard step. ♦

Lemma B.14 *Let* $\bar{\gamma}$ *be a valid marking of* $\mathsf{SLin}^{\star}$ *redexes in* M, *where* $(M, \bar{\gamma}) \xrightarrow[\text{cpl}]{} N$. *Then there exists some term* M_0 *with marking* $\bar{\gamma}_0$ *such that*

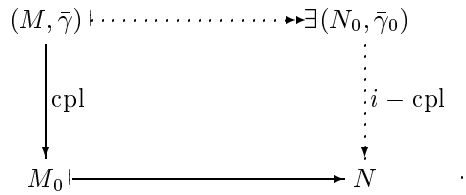
$$\sigma : (M, \bar{\gamma}) \xrightarrow[\mathsf{SLin}^{\star}\text{-dev}]{} (M_0, \bar{\gamma}_0)$$

where $|\sigma|$ *is a standard step* $M \mapsto M_0$ *and* $(M_0, \bar{\gamma}_0) \xrightarrow[\text{cpl}]{}$ N *by internal steps only. Graphically:*



Proof: By FD!($\mathsf{SLin}^{\star}$): we can contract marked standard redexes while they exist, and this sequence must be finite. We then can take the final term to be M_0, and complete the development by the remaining marked steps. By Lemma B.11, this completion is by internal steps only. ♦

Lemma B.15 *Let* $\sigma : (M, \bar{\gamma}) \xrightarrow[i\text{-cpl}]{} M_0$ *and* $M_0 \overset{\delta}{\mapsto} N$ *in* $\mathsf{SLin}^{\star}$. *Then there exists some* $(N_0, \bar{\delta}_0)$ *such that* $\tau : M \mapsto\!\!\!\!\twoheadrightarrow N_0$, $\bar{\gamma}/\tau = \bar{\delta}_0$ *and* $(N_0, \bar{\delta}_0) \xrightarrow[i\text{-cpl}]{} N$. *Graphically:*



Proof: By applying Lemma B.13 inductively for each step in σ, we have a redex at δ_1 whose single residual with respect to σ is δ. Moreover, we know that $\bar{\gamma} \cup \{\delta_1\}$ is again valid since no (δ_1, n) could be an internal step if for some i we have that (δ_1, i) is the standard step. So we can apply FD!($\mathsf{SLin}^{\star}$) and Lemma B.14 for the result. ♦

Lemma B.16 *Let* $M \xrightarrow[(\mathrm{W}!^{i})]{} M_0 \mapsto N$. *Then there exists some* N_0 *such that* $M \mapsto^{=} N_0 \xrightarrow[(\mathrm{W}!^{i})]{} N$.

Proof: The result is a straightforward analysis of the relative positions of the redexes. ♦

Lemma B.17 *Let* $M \xrightarrow[i]{} M_0 \mapsto\!\!\!\!\twoheadrightarrow N$. *Then there exists some* N_0 *such that* $M \mapsto\!\!\!\!\twoheadrightarrow N_0 \xrightarrow[i]{} N$.

Proof: We treat internal ($!^{w}\mathrm{W}$) steps separately from internal contractions by other rules; for the former we have the result simply by induction on the length of the $M_0 \mapsto\!\!\!\!\twoheadrightarrow N$ sequence, with Lemma B.16 at each step. Otherwise, noting first that $M \xrightarrow[i]{} M_0$ is a complete development of a single redex, we proceed by induction on the length of $M_0 \mapsto\!\!\!\!\twoheadrightarrow N$, with Lemma B.15 at each step. ♦

We can now proceed with the main standardization results. We first have a "pseudo-standardization" result for the non-unique sequence produced by $\mathsf{SLin}^{\star}$, and then a full standardization result for SLin.

Proposition B.18 *Let* $M \xrightarrow[\mathsf{SLin}^{\star}]{\!\!\!\!\twoheadrightarrow} A$; *then there exists some* A_0 *such that* $M \xrightarrow[\mathsf{SLin}^{\star}]{\!\!\!\!\twoheadrightarrow} A_0$ *and* $A_0 \xrightarrow[\mathsf{SLin}^{\star}]{\!\!\!\!\twoheadrightarrow} A$.

Proof: By the obvious induction on the number of internal steps in σ, applying Lemma B.17 from the right at each step to partition the reduction sequence into a standard sequence followed by an internal sequence: *i.e.* we have some M_0 such that $M \mapsto\!\!\!\!\twoheadrightarrow M_0 \xrightarrow[i]{\!\!\!\!\twoheadrightarrow} A$. By Lemma B.10, M_0 is in fact another answer, A_0. ♦

Proof of Proposition 2.5: *If* $M \xrightarrow[\mathsf{SLin}]{}\!\!\!\!\!\rightarrow A$ *then there exists some separated-linear lambda answer* A_0 *such that* $M \xmapsto[\mathsf{SLin}]{}\!\!\!\!\!\rightarrow A_0$.

We have that $\xmapsto[\mathsf{SLin}]{}$ selects a unique redex for every (closed) non-answer by Lemma B.1, and that it always finds an answer when one is available by Lemma B.18 with Lemma B.2.