SKInT Labels

Jean Goubault-Larrecq

J. Goubault–Larrecq

SKInT Labels

Research Report LSV–02–7, Jul. 2002

Laboratoire Spécification et Vérification

Ecole Normale Supérieure de Cachan
61, avenue du Président Wilson
94235 Cachan Cedex France

Research Report LSV–02–7, Lab. Spécification et Vérification, CNRS & ENS de Cachan, France, Jul. 2002

SKInT Labels

Jean Goubault-Larrecq

LSV/CNRS UMR 8643, ENS Cachan
61, av. du président-Wilson
94235 Cachan Cedex, France

Abstract. SKIn and SKInT are first-order rewrite systems, i.e., calculi of explicit substitutions in a broad sense, that implement β-reduction in the λ-calculus and are confluent even on open terms. SKInT additionally preserves strong and weak normalization, as well as existence of head normal forms. However it implements call-by-value more naturally than call-by-name. In this paper we investigate a notion of labelling à la Hyland-Wadsworth-Lévy for SKInT. Unsurprisingly, this allows us to reprove that SKInT is confluent. However, the structure of labels is surprisingly more complex than in the λ-calculus, and we try to explain why. Finally, we show that labeled SKInT reduction naturally implements a modified form of labeled reduction in the call-by-value λ-calculus that generalizes superdevelopments à la Aczel.

1 Introduction

Implementing the λ-calculus and, in general, functional languages has long been the subject of research. One line of research, exemplified notably by calculi of explicit substitutions such as $\lambda\sigma$ [1], is to find *first-order* rewrite systems, i.e., without binders such as λ that create opportunities for bugs in implementations. While $\lambda\sigma$ implements β-reduction correctly, it is not terminating in the simply-typed case [12] and is only confluent on so-called semi-closed terms [14]. The question of finding first-order rewrite systems that implement β-reduction correctly (if $u \longrightarrow v$ in the λ-calculus, then $u^* \longrightarrow^+ v^*$ in the given rewrite system, where u^* is some fixed computable translation of λ-terms into the rewrite system), are confluent on open terms, and preserve strong normalization (if u is strongly normalizing in the λ-calculus, then u^* is strongly normalizing in the rewrite system) has long been open. This was first solved independently by Lang and David [6] and by Goguen and the author's SKInT [7]. In particular, the calculus SKInT is confluent on open terms, enjoys a standardization property, terminates on simply-typed terms [7], and preserves strong and weak normalization, as well as existence of head normal forms [8]. The only catch is that the standard translation of λ-terms u to SKInT terms u^* does not map full, call-by-name β-reduction to legal reductions in SKInT. However if $u \longrightarrow v$ in Plotkin's *call-by-value* λ-calculus [13], then $u^* \longrightarrow^+ v^*$ in SKInT, and this can be used to define an alternative, more complex translation $u \mapsto L^*(u)$, obtained by composition with a translation $u \mapsto L(u)$ which maps call-by-name reduction to call-by-value reduction. This results in a conservative embedding of call-by-name calculus inside SKInT. However, it can be rightly said that this is a bit of cheating, and that SKInT naturally encodes, through the $u \mapsto u^*$ translation, something in between call-by-value and call-by-name β-reduction.

It has been argued [5] that this intermediate notion of reduction might in fact be an alternative definition of call-by-value, with better logical properties:

- *SKInT reduction simulates, but is not Plotkin's call-by-value reduction*. While SKInT implements call-by-value, it actually implements more. For instance, $((\lambda x \cdot x)(yz))^*$ reduces to $(yz)^*$ in SKInT, while $(\lambda x \cdot x)(yz)$ is normal in Plotkin's call-by-value λ-calculus. So the notion of reduction in SKInT is naturally strictly in between Plotkin's call-by-value and call-by-name reduction.
- *SKInT has good logical properties*. SKInT arises as a language of proof terms, via the Curry-Howard isomorphism, for a natural logic. This logic, *near-intuitionistic logic*, is characterized by Kripke frames where the accessibility relation is a pre-order, just like intuitionistic logic, except the set of worlds where atomic formulas hold are not restricted to be upper sets [7]. Alternatively, this can be seen as a fragment of the modal logic S4, through the translation of (near-)intuitionistic implication $F \supset G$ as $\Box(F \Rightarrow G)$, where $\Rightarrow$ is intuitionistic or classical implication.

Note that Plotkin's call-by-value λ-calculus can be analyzed through Moggi's meta-language [4], which corresponds to intuitionistic *lax logic*. No first-order calculus implementing Moggi's meta-language is known.

We seek here to understand the notion of call-by-value reduction that SKInT offers by finding and studying a satisfactory notion of *labeled reduction*, such as those introduced in the λ-calculus by Hyland, Wadsworth and Lévy [9, 15, 11]. There are a number of ways we can justify our choice of labels and our label calculus. We would have loved to explain our labels as abstract representations of *paths*, as in [2]. However there are at least two difficulties here. The first is that paths are a way of connecting principal ports of λ operators to principal ports of applications, *regardless* of any call-by-value related concern. The second—and this is more complicated to explain—is that we cannot talk about paths in SKInT, rather we need to talk about (infinite) sequences of paths. This is related to the fact that a SKInT term M does not just have one translation $[\![M]\!]$ as a λ-term (see [7]), but as many translations $u_1, \ldots, u_n \mapsto [\![M]\!]\,(u_1, \ldots, u_n)$, one for each natural number n [7]. This implies in particular that our labels will be *infinite sequences* of Lévy labels.

Outline. We recapitulate what SKInT is, and how it relates to the λ-calculus, in Section 2. Instead of introducing SKInT labels as notations for paths, we use the fact that the correct notion of path should be the one of the λ-calculus, lifted to SKInT along the standard translation $u_1, \ldots, u_n \mapsto [\![M]\!]\,(u_1, \ldots, u_n)$ (see [7]), and we reverse-engineer λ-calculus labels to the sought after labels for SKInT. We explore this in Section 3, and show that using Hyland-Wadsworth labels, where contracting redexes consume finite resources, yield a terminating labeled variant of SKInT. This is as in the λ-calculus. To be fair, we use a modified, infinitely η-expanded version of the λ-calculus with labels and an additional $\oplus$ operator. We then produce a new proof of the fact that SKInT is confluent in Section 4. Reverse-engineering SKInT labels into call-by-value λ-calculus yields a slightly different notion of labels for the λ-calculus: whereas Hyland-Wadsworth labels generalize the finite developments theorem, ours generalize superdevelopments [10]. This is shown in Section 5. We conclude in Section 6.

2 Preliminaries on SKInT

Recall that the syntax of the λ-calculus is [3]:

$$t ::= x \mid tt \mid \lambda x \cdot t$$

where x ranges over an infinite set of so-called variables, and terms s and t that are α-equivalent are considered equal; we denote λ-terms by $s, t, \ldots$, and variables by x, y, z, etc. We shall write $=$ for α-equivalence; in the first-order calculi to come, $=$ will denote syntactic equality.

The basic computation rule is β-reduction, the compatible closure of:

$$(\beta) \quad (\lambda x \cdot t)s \to t[x := s]$$

where $t[x := s]$ denotes capture-avoiding substitution. The relation $\longrightarrow$ is the compatible closure of this relation, $\longrightarrow^*$ is the reflexive-transitive closure of the latter, and $\longrightarrow^+$ is its transitive closure. We shall use $\longrightarrow$, $\longrightarrow^*$, $\longrightarrow^+$ ambiguously in other calculi as well, taking care to make clear which is intended. Plotkin's (β_V) rule for *call-by-value λ-calculus* [13] is (β) restricted to the cases where s is a value, that is, a variable or a λ-abstraction.

The terms of SKInT, and of its companion calculus SKIn [7], on the other hand, are defined by the grammar:

$$M ::= x \mid \mathrm{I}_\ell \mid \mathrm{S}_\ell\,(M, M) \mid \mathrm{K}_\ell\,(M)$$

where ℓ ranges over $\mathbb{N}$. This is an infinitary first-order language. The reduction rules of SKInT are shown in Figure 1, thus defining an infinite rewrite system. The semantical idea behind SKInT, or SKIn, will be made clear by stating an informal translation from SKInT (or SKIn) to the λ-calculus. Informally:

$$\begin{aligned} \mathrm{I}_\ell &\sim \lambda x_0 \cdot \ldots \cdot \lambda x_{\ell-1} \cdot \lambda x_\ell \cdot x_\ell \\ \mathrm{S}_\ell\,(M, N) &\sim \lambda x_0 \cdot \ldots \cdot \lambda x_{\ell-1} \cdot M x_0 \ldots x_{\ell-1}(N x_0 \ldots x_{\ell-1}) \\ \mathrm{K}_\ell\,(M) &\sim \lambda x_0 \cdot \ldots \cdot \lambda x_{\ell-1} \cdot \lambda x_\ell \cdot M x_0 \ldots x_{\ell-1} \end{aligned}$$

So $\mathrm{I}_\ell, \mathrm{S}_\ell, \mathrm{K}_\ell$ generalize Curry's combinators I, S and K respectively.

SKIn is defined as SKInT, except that rule $(\mathrm{K}_\ell \mathrm{S}_{\mathcal{L}+1})$ is replaced by $(\mathrm{K}_\ell \mathrm{S}_{\mathcal{L}})$: $\mathrm{K}_\ell\,(\mathrm{S}_{\mathcal{L}-1}\,(M, N)) \to \mathrm{S}_{\mathcal{L}}\,(\mathrm{K}_\ell\,(M), \mathrm{K}_\ell\,(N))$; conversely, SKInT is as SKIn, except that rule $(\mathrm{K}_\ell \mathrm{S}_{\mathcal{L}})$ is restricted to the case $\ell < \mathcal{L} - 1$.

We can split SKInT in two: the set of all rules (SI_ℓ), $\ell \geq 0$, corresponds somehow to the actual β-reduction rule of the λ-calculus, or more precisely to βI-reduction (the notion of reduction of λI), and we shall call this group of rules βI. All other rules essentially correspond to the propagation of substitutions in the λ-calculus, and we call the set of these rules ΣT. Similarly, Σ is SKIn minus βI. It turns out that both Σ and ΣT are confluent, but ΣT terminates while Σ only normalizes weakly (even in a typed setting, see [7]).

The natural translation from the λ-calculus to SKInT, resp. SKIn, is $t \mapsto t^*$, defined in Figure 2. Whenever $u \longrightarrow v$ in the λ-calculus, $u^* \longrightarrow^+ v^*$ in SKIn, but not in SKInT. Still, $u^* \longrightarrow^+ v^*$ in SKInT as soon as $u \longrightarrow v$ in Plotkin's call-by-value λ-calculus. We shall refine this observation in Section 5.

$$(\mathrm{SI}_\ell)\quad \mathrm{S}_\ell\,(\mathrm{I}_\ell, P) \to P \qquad (\mathrm{SK}_\ell)\quad \mathrm{S}_\ell\,(\mathrm{K}_\ell\,(M)\,, P) \to M$$

$$\begin{array}{lll}
(\mathrm{S}_\ell\mathrm{I}_\mathcal{L}) & \mathrm{S}_\ell\,(\mathrm{I}_\mathcal{L}, P) \to \mathrm{I}_{\mathcal{L}-1} & (\mathrm{K}_\ell\mathrm{I}_\mathcal{L}) \quad \mathrm{K}_\ell\,(\mathrm{I}_{\mathcal{L}-1}) \to \mathrm{I}_\mathcal{L} \\
(\mathrm{S}_\ell\mathrm{K}_\mathcal{L}) & \mathrm{S}_\ell\,(\mathrm{K}_\mathcal{L}\,(M)\,, P) \to \mathrm{K}_{\mathcal{L}-1}\,(\mathrm{S}_\ell\,(M, P)) & (\mathrm{K}_\ell\mathrm{K}_\mathcal{L}) \quad \mathrm{K}_\ell\,(\mathrm{K}_{\mathcal{L}-1}\,(M)) \to \mathrm{K}_\mathcal{L}\,(\mathrm{K}_\ell\,(M)) \\
(\mathrm{S}_\ell\mathrm{S}_\mathcal{L}) & \mathrm{S}_\ell\,(\mathrm{S}_\mathcal{L}\,(M, N)\,, P) & (\mathrm{K}_\ell\mathrm{S}_{\mathcal{L}+1}) \quad \mathrm{K}_\ell\,(\mathrm{S}_\mathcal{L}\,(M, P)) \\
& \quad \to \mathrm{S}_{\mathcal{L}-1}\,(\mathrm{S}_\ell\,(M, P)\,, \mathrm{S}_\ell\,(N, P)) & \quad \to \mathrm{S}_{\mathcal{L}+1}\,(\mathrm{K}_\ell\,(M)\,, \mathrm{K}_\ell\,(P))
\end{array}$$

Fig. 1. SKInT reduction rules (for every $0 \leq \ell < \mathcal{L}$)

$$\begin{array}{lll}
x^* & = x & [x]x & = \mathrm{I}_0 \\
(st)^* & = \mathrm{S}_0\,(s^*, t^*) & [x]y & = \mathrm{K}_0\,(y) \qquad (y \neq x) \\
(\lambda x \cdot t)^* & = [x](t^*) & [x](\mathrm{I}_\ell) & = \mathrm{I}_{\ell+1} \\
 & & [x](\mathrm{S}_\ell\,(M, N)) & = \mathrm{S}_{\ell+1}\,([x]M, [x]N) \\
 & & [x](\mathrm{K}_\ell\,(M)) & = \mathrm{K}_{\ell+1}\,([x]M)
\end{array}$$

Fig. 2. Translation from the λ-calculus to SKIn

3 Introducing SKInT labels

Recall the intuitive meaning of SKInT terms as λ-terms, e.g., $\mathrm{I}_\ell \sim \lambda x_0 \cdot \ldots \cdot \lambda x_{\ell-1} \cdot \lambda x_\ell \cdot x_\ell$. This can be made into a translation of SKInT to the λ-calculus. However, doing this maps SKInT reductions to λ-reductions involving *both* β and η (see e.g. Figure 2, Theorem 4.13, or Theorem 5.3 in [7]), and labeled reduction does not mix well with η-reduction. There is an easy way out: deal with infinitely η-expanded terms instead, e.g., $\mathrm{I}_\ell \sim \lambda x_0 \cdot \ldots \cdot \lambda x_{\ell-1} \cdot \lambda x_\ell \cdot \lambda x_{\ell+1} \cdot \ldots \cdot \lambda x_k \cdot \ldots \cdot x_\ell x_{\ell+1} \ldots x_k \ldots$. We add a few gadgets like the infix binary operator $\oplus$ (a technical device introduced in [7], Theorem 4.13 to help in strong normalization proofs), and labels, which decorate λ-abstraction and variables, to be introduced next.

Call a *label structure* any 4-tuple $(\mathcal{M}, 1, \bullet, m \mapsto \underline{m})$, where $(\mathcal{M}, 1, \bullet)$ is a monoid. Let $\boldsymbol{x}$ denote any infinite sequence of *pairwise distinct* variables $x_0, x_1, \ldots$; similarly, let $\boldsymbol{m}$ denote any infinite sequence of (possibly identical) labels $m_0, m_1, \ldots$ *Labeled variables* x^m are pairs of a variable and a label. This is extended to infinite sequences $\boldsymbol{x^m}$, defined by $(\boldsymbol{x^m})_i \hat{=} x_i^{m_i}$. The syntax of the λ_η^∞-calculus is given by:

$$\begin{array}{lll}
s, t ::= s \oplus t \mid \lambda^m \boldsymbol{x} \cdot e \mid x^m & & \text{Terms} \\
e ::= s(\boldsymbol{t}) & & \text{Bodies} \\
\boldsymbol{t} ::= \boldsymbol{x^m} \mid s, \boldsymbol{t} & & \text{Argument lists}
\end{array}$$

where $s, \boldsymbol{t}$ denotes the infinite sequence whose first element is s, and whose remaining elements are those of $\boldsymbol{t}$. Note that λ-abstractions $\lambda^m \boldsymbol{x} \cdot e$ bind infinitely many variables at once, are decorated with a single label m, and their bodies $e = s(\boldsymbol{t})$ apply some *head* term s to some infinite *argument list* $\boldsymbol{t}$.

The rule $(\oplus-)$ below shows that $s \oplus t$ behaves essentially like t, so we might be tempted by just writing t instead of $s \oplus t$. In fact, doing so would also allow us to define

a label calculus for SKIn, too, but then termination of labeled reduction à la Hyland-Wadsworth (Theorem 2 in the case of SKInT) would be lost. By the way, translations using $\oplus$ only work for SKInT, not for SKIn, so instead of dismissing $\oplus$ as a technical artifact, we take it seriously as an indication of what the specificities of call by value should be.

Labels act on λ_η^∞-terms on the right by:

$$(s \oplus t)^m \hat{=} s \oplus (t)^m \qquad (\lambda^{m_0}\boldsymbol{x} \cdot e)^m \hat{=} \lambda^{m_0 \bullet m}\boldsymbol{x} \cdot e \qquad (x^{m_0})^m \hat{=} x^{m_0 \bullet m}$$

(*Acting on the right* means that $(t)^1 = t$, and $(t)^{m \bullet m'} = ((t)^m)^{m'}$. This is easily checked.)

This allows us to define a notion of substitution $s[\boldsymbol{x} := \boldsymbol{t}]$ of infinitely many variables $\boldsymbol{x}$ by infinitely many terms in the argument list $\boldsymbol{t}$ in s in a mostly obvious way:

$$(x^m)[\boldsymbol{x} := \boldsymbol{t}] \hat{=} x^m \text{ (if } x \neq x_i \text{ for all } i) \qquad (x^m)[\boldsymbol{x} := \boldsymbol{t}] \hat{=} (t_i)^m \text{ (if } x = x_i \text{ for some } i)$$
$$(s \oplus t)[\boldsymbol{x} := \boldsymbol{t}] \hat{=} s[\boldsymbol{x} := \boldsymbol{t}] \oplus t[\boldsymbol{x} := \boldsymbol{t}] \qquad (\lambda^m \boldsymbol{y} \cdot e)[\boldsymbol{x} := \boldsymbol{t}] \hat{=} \lambda^m \boldsymbol{y} \cdot (e[\boldsymbol{x} := \boldsymbol{t}])$$
$$(s(\boldsymbol{u}))[\boldsymbol{x} := \boldsymbol{t}] \hat{=} s[\boldsymbol{x} := \boldsymbol{t}](\boldsymbol{u}[\boldsymbol{x} := \boldsymbol{t}])$$

where in the $\lambda^m \boldsymbol{y} \cdot e$ case, $x_i \neq y_j$ for all i, j and no y_i is free in $\boldsymbol{t}$, and where $\boldsymbol{u}[\boldsymbol{x} := \boldsymbol{t}]$ is defined by $(\boldsymbol{u}[\boldsymbol{x} := \boldsymbol{t}])_i \hat{=} u_i[\boldsymbol{x} := \boldsymbol{t}]$. It is easy to check that $\boldsymbol{u}[\boldsymbol{x} := \boldsymbol{t}]$ is an argument list again. The λ_η^∞-terms are defined up to *infinite α-equivalence*, which allows for the replacement of infinitely many bound variables $\boldsymbol{x}$ by $\boldsymbol{y}$ at once. This allows us to make substitution total modulo α. We shall always reason modulo infinite α-equivalence without saying so explicitly.

The *top label* $\mathrm{top}(t)$ of t is defined by:

$$\mathrm{top}(s \oplus t) \hat{=} \mathrm{top}(t) \qquad \mathrm{top}(\lambda^m \boldsymbol{x} \cdot e) \hat{=} m \qquad \mathrm{top}(x^m) \hat{=} m$$

For each label $m \in \mathcal{M}$, let m^ω be the infinite list of labels whose sole element is m. Define *labeled reduction* on the λ_η^∞-calculus by the rules:

$$(\beta^\omega)\ (\lambda^m \boldsymbol{x} \cdot e)(\boldsymbol{t}) \to e[\boldsymbol{x} := (\boldsymbol{t})^{\overline{m}^\omega}]$$
$$(\oplus -)\ s \oplus t \to t \qquad (\oplus)\ (s \oplus t) \oplus u \to s \oplus (t \oplus u)$$

The important rule is (β^ω). Note the similarity with Lévy labeled reduction $(\lambda^m x \cdot e)(t) \to (e[x := (t)^{\overline{m}}])^{\overline{m}}$. The differences are that, first, infinitely many variables x_i are replaced by infinitely many terms $(t_i)^{\overline{m}}$ in one step, and second and most importantly, the labeling of the whole reduct by $\overline{m}$ simply disappears. The reason is that redexes are never created upwards in the λ_η^∞-calculus: by syntactic restrictions, the right-hand side $e[\boldsymbol{x} := (\boldsymbol{t})^{\overline{m}^\omega}]$ is a *body*, and can therefore never be applied to any argument list.

A distinguishing feature of labeled reduction in the λ-calculus is that if labeled β-reduction is restricted to labels with at most p nested underlinings, with p fixed, then every such *p-bounded* labeled reduction terminates (local strong normalization [3]). We shall prove this for λ_η^∞ in Section 3.2, and then for labeled SKInT in Section 3.3.

Now, provided we ignore labels, there is a translation $[\![\_]\!]_\omega$ of SKInT into λ_η^∞ that preserves reduction; this is basically the $[\![\_]\!]_\bullet$ translation of [7], Figure 8. This translation

is as follows, where we have not yet indicated labels. We take the convention that $\oplus$ associates on the right, so $t_0 \oplus t_1 \oplus \ldots \oplus t_{\ell-1} \oplus t_\ell$ denotes $t_0 \oplus (t_1 \oplus (\ldots \oplus (t_{\ell-1} \oplus t_\ell) \ldots))$ if $\ell \geq 1$, t_ℓ if $\ell = 0$. We also use the abbreviation $\boldsymbol{t}_{i..j}$ for the sequence $t_i, \ldots, t_j$, and similarly $\boldsymbol{t}_{i..\infty}$ for $t_i, t_{i+1}, \ldots$:

$$\begin{array}{c} [\![x]\!]_\omega(\boldsymbol{t}) \mathrel{\hat{=}} x(\boldsymbol{t}) \\ [\![\mathrm{I}_\ell]\!]_\omega(\boldsymbol{t}) \mathrel{\hat{=}} (t_0 \oplus \ldots \oplus t_{\ell-1} \oplus t_\ell)(\boldsymbol{t}_{\ell+1..\infty}) \\ [\![\mathrm{S}_\ell(M,N)]\!]_\omega(\boldsymbol{t}) \mathrel{\hat{=}} [\![M]\!]_\omega(\boldsymbol{t}_{0..\ell-1}, \lambda \boldsymbol{x} \cdot [\![N]\!]_\omega(\boldsymbol{t}_{0..\ell-1}, \boldsymbol{x}), \boldsymbol{t}_{\ell..\infty}) \\ [\![\mathrm{K}_\ell(N)]\!]_\omega(\boldsymbol{t}) \mathrel{\hat{=}} [\![M]\!]_\omega(\boldsymbol{t}_{0..\ell-1}, t_\ell \oplus t_{\ell+1}, \boldsymbol{t}_{\ell+2..\infty}) \end{array}$$

(Let us stress that this is yet provisional; the real translation will be given in Figure 3.)

Let us now reverse-engineer labels for SKInT, so that labeled SKInT-reduction maps to λ_η^∞-reduction. The translation above gives a view of each SKInT term M as a map $[\![M]\!]_\omega$ from an infinite sequence $\boldsymbol{t}$ of λ_η^∞-terms to some λ_η^∞-term. Labels should allow us to trace each element t_i of $\boldsymbol{t}$ through reductions in $[\![M]\!]_\omega(\boldsymbol{t})$. This invites us to define SKInT labels as *infinite sequences* $\boldsymbol{m}$ of labels, one m_i for each t_i.

Let therefore $\mathcal{M}^\omega$ be the set of all infinite sequences of labels in $\mathcal{M}$. Let $\boldsymbol{m} \bullet \boldsymbol{m}'$ be the sequence of all labels $m_i \bullet m'_i$, and $\underline{\boldsymbol{m}}$ be the sequence of all labels $\underline{m_i}$, $i \in \mathbb{N}$. This defines a new label structure $(\mathcal{M}^\omega, 1^\omega, \bullet, \boldsymbol{m} \mapsto \underline{\boldsymbol{m}})$. Given this, we would like to define an action of $\mathcal{M}^\omega$ on SKInT (or rather, labeled SKInT) terms mapping $\boldsymbol{m}, M$ to ${}^{\boldsymbol{m}}(M)$. The natural way to trace each t_i through reductions in $[\![M]\!]_\omega(\boldsymbol{t})$ is by considering $[\![M]\!]_\omega((\boldsymbol{t})^{\boldsymbol{m}})$, where $(\boldsymbol{t})^{\boldsymbol{m}}$ denotes the infinite sequence $(t_0)^{m_0}, (t_1)^{m_1}, \ldots$ The action ${}^{\boldsymbol{m}}(M)$ should allow us to do this tracing job in SKInT, in other words $[\![M]\!]_\omega((\boldsymbol{t})^{\boldsymbol{m}})$ should be exactly the same as $[\![{}^{\boldsymbol{m}}(M)]\!]_\omega(\boldsymbol{t})$. This will be Lemma 2.

3.1 The Definition of SKInT Labels

It turns out that a simple way to do this is to add SKInT labels to free variables x and to the constants I_ℓ only. We therefore get the following modified syntax for *labeled SKInT terms*:

$$M ::= x^{\boldsymbol{m}} \mid \mathrm{I}_\ell^{\boldsymbol{m}} \mid \mathrm{S}_\ell(M,M) \mid \mathrm{K}_\ell(M)$$

Define the action of $\mathcal{M}^\omega$ on labeled SKInT terms (on the *left*) by:

$$\begin{array}{cc} {}^{\boldsymbol{m}}\left(x^{\boldsymbol{m}'}\right) \mathrel{\hat{=}} x^{\boldsymbol{m} \bullet \boldsymbol{m}'} & {}^{\boldsymbol{m}}\left(\mathrm{I}_\ell^{\boldsymbol{m}'}\right) \mathrel{\hat{=}} \mathrm{I}_\ell^{\boldsymbol{m}_{\ell..\infty} \bullet \boldsymbol{m}'} \\ {}^{\boldsymbol{m}}(\mathrm{S}_\ell(M,N)) \mathrel{\hat{=}} \mathrm{S}_\ell\left({}^{\delta_\ell(\boldsymbol{m})}(M), {}^{\chi_\ell(\boldsymbol{m})}(N)\right) & {}^{\boldsymbol{m}}(\mathrm{K}_\ell(M)) \mathrel{\hat{=}} \mathrm{K}_\ell\left({}^{\sigma_\ell(\boldsymbol{m})}(M)\right) \end{array}$$

where:

$$\begin{array}{cc} \delta_\ell(\boldsymbol{m}) \mathrel{\hat{=}} \boldsymbol{m}_{0..\ell-1}, 1, \boldsymbol{m}_{\ell..\infty} & \chi_\ell(\boldsymbol{m}) \mathrel{\hat{=}} \boldsymbol{m}_{0..\ell-1}, 1^\omega \\ \multicolumn{2}{c}{\sigma_\ell(\boldsymbol{m}) \mathrel{\hat{=}} \boldsymbol{m}_{0..\ell-1}, \boldsymbol{m}_{\ell+1..\infty}} \end{array}$$

This is indeed an action:

Lemma 1 (Action). *For every labeled SKInT term M, ${}^{1^\omega}(M) = M$; for every $\boldsymbol{m}, \boldsymbol{m}' \in \mathcal{M}^\omega$, ${}^{\boldsymbol{m} \bullet \boldsymbol{m}'}(M) = {}^{\boldsymbol{m}}\left({}^{\boldsymbol{m}'}(M)\right)$.*

Proof. Easy structural induction on M. □

We can now formally define the $[\![\text{-}]\!]_\omega$ translation, including all needed labels. This requires $\mathcal{M}$ has a *right zero* 0, i.e., $m \bullet 0 = 0$ for every $m \in \mathcal{M}$. We can always adjoin one freely to $\mathcal{M}$ if there is none already in $\mathcal{M}$. See Figure 3. We take $\lambda \boldsymbol{x} \cdot e$ as an abbreviation of $\lambda^1 \boldsymbol{x} \cdot e$; single variables x denote x^1, and $\boldsymbol{x}$ in argument lists stands for $(\boldsymbol{x})^{1^\omega}$.

$$
\begin{aligned}
[\![x^{\boldsymbol{m}}]\!]_\omega(\boldsymbol{t}) &\mathrel{\hat{=}} x((\boldsymbol{t})^{\boldsymbol{m}}) \\
[\![\mathrm{I}_\ell^{\boldsymbol{m}}]\!]_\omega(\boldsymbol{t}) &\mathrel{\hat{=}} ((t_0)^0 \oplus \ldots \oplus (t_{\ell-1})^0 \oplus (t_\ell)^{m_0})((\boldsymbol{t}_{\ell+1..\infty})^{\boldsymbol{m}_{1..\infty}}) \\
[\![\mathrm{S}_\ell(M,N)]\!]_\omega(\boldsymbol{t}) &\mathrel{\hat{=}} [\![M]\!]_\omega(\boldsymbol{t}_{0..\ell-1},\ \lambda \boldsymbol{x} \cdot [\![N]\!]_\omega(\boldsymbol{t}_{0..\ell-1}, \boldsymbol{x}),\ \boldsymbol{t}_{\ell..\infty}) \\
[\![\mathrm{K}_\ell(N)]\!]_\omega(\boldsymbol{t}) &\mathrel{\hat{=}} [\![M]\!]_\omega(\boldsymbol{t}_{0..\ell-1},\ (t_\ell)^0 \oplus t_{\ell+1},\ \boldsymbol{t}_{\ell+2..\infty})
\end{aligned}
$$

Fig. 3. Translation of labeled SKInT terms to λ_η^∞-terms

Lemma 2. *For every labeled SKInT term M, for every $\boldsymbol{m} \in \mathcal{M}^\omega$, for every argument list $\boldsymbol{t}$, $[\![^{\boldsymbol{m}}(M)]\!]_\omega(\boldsymbol{t}) = [\![M]\!]_\omega((\boldsymbol{t})^{\boldsymbol{m}})$.*

Proof. By structural induction on M. If $M = x^{\boldsymbol{m}'}$, $[\![^{\boldsymbol{m}}(M)]\!]_\omega(\boldsymbol{t}) = \left[\!\!\left[x^{\boldsymbol{m} \bullet \boldsymbol{m}'}\right]\!\!\right]_\omega(\boldsymbol{t}) = x((\boldsymbol{t})^{\boldsymbol{m} \bullet \boldsymbol{m}'}) = x(((\boldsymbol{t})^{\boldsymbol{m}})^{\boldsymbol{m}'}) = [\![M]\!]_\omega((\boldsymbol{t})^{\boldsymbol{m}})$.

If $M = \mathrm{I}_\ell^{\boldsymbol{m}'}$, $[\![^{\boldsymbol{m}}(M)]\!]_\omega(\boldsymbol{t}) = \left[\!\!\left[\mathrm{I}_\ell^{\boldsymbol{m}_{\ell..\infty} \bullet \boldsymbol{m}'}\right]\!\!\right]_\omega(\boldsymbol{t}) = ((t_0)^0 \oplus \ldots \oplus (t_{\ell-1})^0 \oplus (t_\ell)^{m_\ell \bullet m'_0})((\boldsymbol{t}_{\ell+1..\infty})^{\boldsymbol{m}_{\ell+1..\infty} \bullet \boldsymbol{m}'_{1..\infty}}) = ((t_0)^{m_0 \bullet 0} \oplus \ldots \oplus (t_{\ell-1})^{m_{\ell-1} \bullet 0} \oplus (t_\ell)^{m_\ell \bullet m'_0})((\boldsymbol{t}_{\ell+1..\infty})^{\boldsymbol{m}_{\ell+1..\infty} \bullet \boldsymbol{m}'_{1..\infty}}) = [\![M]\!]_\omega((\boldsymbol{t})^{\boldsymbol{m}})$.

If $M = \mathrm{S}_\ell(N,P)$, $[\![^{\boldsymbol{m}}(M)]\!]_\omega(\boldsymbol{t}) = \left[\!\!\left[\mathrm{S}_\ell\left({}^{\delta_\ell(\boldsymbol{m})}(N), {}^{\chi_\ell(\boldsymbol{m})}(P)\right)\right]\!\!\right]_\omega(\boldsymbol{t}) = \left[\!\!\left[{}^{\delta_\ell(\boldsymbol{m})}(N)\right]\!\!\right]_\omega(\boldsymbol{t}_{0..\ell-1},\ \lambda \boldsymbol{x} \cdot \left[\!\!\left[{}^{\chi_\ell(\boldsymbol{m})}(P)\right]\!\!\right]_\omega(\boldsymbol{t}_{0..\ell-1}, \boldsymbol{x}),\ \boldsymbol{t}_{\ell..\infty}) = \left[\!\!\left[{}^{\delta_\ell(\boldsymbol{m})}(N)\right]\!\!\right]_\omega(\boldsymbol{t}_{0..\ell-1},\ \lambda \boldsymbol{x} \cdot [\![P]\!]_\omega((\boldsymbol{t}_{0..\ell-1})^{\chi_\ell(\boldsymbol{m})_{0..\ell-1}}, (\boldsymbol{x})^{\chi_\ell(\boldsymbol{m})_{\ell..\infty}}),\ \boldsymbol{t}_{\ell..\infty})$ (by induction hypothesis) $= \left[\!\!\left[{}^{\delta_\ell(\boldsymbol{m})}(N)\right]\!\!\right]_\omega(\boldsymbol{t}_{0..\ell-1},\ \lambda \boldsymbol{x} \cdot [\![P]\!]_\omega((\boldsymbol{t}_{0..\ell-1})^{\boldsymbol{m}_{0..\ell-1}}, \boldsymbol{x}),\ \boldsymbol{t}_{\ell..\infty})$ (by definition of χ_ℓ) $= [\![N]\!]_\omega((\boldsymbol{t}_{0..\ell-1})^{\delta_\ell(\boldsymbol{m})_{0..\ell-1}}, (\lambda \boldsymbol{x} \cdot [\![P]\!]_\omega((\boldsymbol{t}_{0..\ell-1})^{\boldsymbol{m}_{0..\ell-1}}, \boldsymbol{x}))^{\delta_\ell(\boldsymbol{m})_\ell}, (\boldsymbol{t}_{\ell..\infty})^{\delta_\ell(\boldsymbol{m})_{\ell+1..\infty}})$ (by induction hypothesis) $= [\![N]\!]_\omega((\boldsymbol{t}_{0..\ell-1})^{\boldsymbol{m}_{0..\ell-1}}, \lambda^1 \boldsymbol{x} \cdot [\![P]\!]_\omega((\boldsymbol{t}_{0..\ell-1})^{\boldsymbol{m}_{0..\ell-1}}, \boldsymbol{x}), (\boldsymbol{t}_{\ell..\infty})^{\boldsymbol{m}_{\ell..\infty}})$ (by definition of δ_ℓ) $= [\![M]\!]_\omega((\boldsymbol{t})^{\boldsymbol{m}})$.

If $M = \mathrm{K}_\ell(N)$, $[\![^{\boldsymbol{m}}(M)]\!]_\omega(\boldsymbol{t}) = \left[\!\!\left[\mathrm{K}_\ell\left({}^{\sigma_\ell(\boldsymbol{m})}(N)\right)\right]\!\!\right]_\omega(\boldsymbol{t}) = \left[\!\!\left[{}^{\sigma_\ell(\boldsymbol{m})}(N)\right]\!\!\right]_\omega(\boldsymbol{t}_{0..\ell-1},\ (t_\ell)^0 \oplus t_{\ell+1},\ \boldsymbol{t}_{\ell+2..\infty}) = [\![N]\!]_\omega((\boldsymbol{t}_{0..\ell-1})^{\sigma_\ell(\boldsymbol{m})_{0..\ell-1}}, ((t_\ell)^0 \oplus t_{\ell+1})^{\sigma_\ell(\boldsymbol{m})_\ell}, (\boldsymbol{t}_{\ell+2..\infty})^{\sigma_\ell(\boldsymbol{m})_{\ell+1..\infty}})$ (by induction hypothesis) $= [\![N]\!]_\omega((\boldsymbol{t}_{0..\ell-1})^{\boldsymbol{m}_{0..\ell-1}},\ (t_\ell)^0 \oplus (t_{\ell+1})^{m_{\ell+1}},\ (\boldsymbol{t}_{\ell+2..\infty})^{\boldsymbol{m}_{\ell+2..\infty}})$ (by definition of σ_ℓ) $= [\![N]\!]_\omega((\boldsymbol{t}_{0..\ell-1})^{\boldsymbol{m}_{0..\ell-1}}, (t_\ell)^{m_\ell \bullet 0} \oplus (t_{\ell+1})^{m_{\ell+1}}, (\boldsymbol{t}_{\ell+2..\infty})^{\boldsymbol{m}_{\ell+2..\infty}}) = [\![\mathrm{K}_\ell(N)]\!]_\omega((\boldsymbol{t})^{\boldsymbol{m}})$. □

Let us now reverse-engineer how labels should be handled in SKInT reduction. Let us look at the βI-redex $\mathrm{S}_\ell\,(\mathrm{I}_\ell^{\boldsymbol{m}}, P)$:

$$\begin{aligned}
[\![\mathrm{S}_\ell\,(\mathrm{I}_\ell^{\boldsymbol{m}}, P)]\!]_\omega\,(\boldsymbol{t}) &= [\![\mathrm{I}_\ell^{\boldsymbol{m}}]\!]_\omega\,(\boldsymbol{t}_{0..\ell-1},\ \lambda\boldsymbol{x}\cdot[\![P]\!]_\omega\,(\boldsymbol{t}_{0..\ell-1},\boldsymbol{x}),\ \boldsymbol{t}_{\ell..\infty}) \\
&= ((t_0)^0 \oplus \ldots \oplus (t_{\ell-1})^0 \oplus (\lambda\boldsymbol{x}\cdot[\![P]\!]_\omega\,(\boldsymbol{t}_{0..\ell-1},\boldsymbol{x}))^{m_0})((\boldsymbol{t}_{\ell..\infty})^{\boldsymbol{m}_{1..\infty}}) \\
&\longrightarrow^* (\lambda^{m_0}\boldsymbol{x}\cdot[\![P]\!]_\omega\,(\boldsymbol{t}_{0..\ell-1},\boldsymbol{x})((\boldsymbol{t}_{\ell..\infty})^{\boldsymbol{m}_{1..\infty}}) \\
&\longrightarrow [\![P]\!]_\omega\,(\boldsymbol{t}_{0..\ell-1},\boldsymbol{x})[\boldsymbol{x} := ((\boldsymbol{t}_{\ell..\infty})^{\boldsymbol{m}_{1..\infty}})^{\underline{m_0}}] \qquad (1)
\end{aligned}$$

by (β^ω). Now it is easy to see that the latter equals exactly

$$[\![P]\!]_\omega\,(\boldsymbol{t}_{0..\ell-1}, (\boldsymbol{t}_{\ell..\infty})^{\boldsymbol{m}_{1..\infty}\bullet\underline{m_0}^\omega})$$

By Lemma 2, this is exactly $\left[\!\!\left[1^\ell, (\boldsymbol{m}_{1..\infty}\bullet\underline{m_0}^\omega)\right]\!\!\right]_\omega(\boldsymbol{t})$, where 1^ℓ denotes ℓ occurrences of 1. The reduction rules of the labeled SKInT calculus follow by the same token. See Figure 4.

$$(\mathrm{SI}_\ell^{\boldsymbol{m}})\quad \mathrm{S}_\ell\,(\mathrm{I}_\ell^{\boldsymbol{m}}, P) \to {}^{1^\ell,(\boldsymbol{m}_{1..\infty}\bullet\underline{m_0}^\omega)}(P) \qquad (\mathrm{SK}_\ell)\quad \mathrm{S}_\ell\,(\mathrm{K}_\ell\,(M), P) \to M$$

$$\begin{array}{ll}
(\mathrm{S}_\ell\mathrm{I}_{\mathcal{L}}^{\boldsymbol{m}})\ \mathrm{S}_\ell\,(\mathrm{I}_{\mathcal{L}}^{\boldsymbol{m}}, P) \to \mathrm{I}_{\mathcal{L}-1}^{\boldsymbol{m}} & (\mathrm{K}_\ell\mathrm{I}_{\mathcal{L}}^{\boldsymbol{m}})\quad \mathrm{K}_\ell\,(\mathrm{I}_{\mathcal{L}-1}^{\boldsymbol{m}}) \to \mathrm{I}_{\mathcal{L}}^{\boldsymbol{m}} \\
(\mathrm{S}_\ell\mathrm{K}_{\mathcal{L}})\ \mathrm{S}_\ell\,(\mathrm{K}_{\mathcal{L}}\,(M), P) \to \mathrm{K}_{\mathcal{L}-1}\,(\mathrm{S}_\ell\,(M,P)) & (\mathrm{K}_\ell\mathrm{K}_{\mathcal{L}})\quad \mathrm{K}_\ell\,(\mathrm{K}_{\mathcal{L}-1}\,(M)) \to \mathrm{K}_{\mathcal{L}}\,(\mathrm{K}_\ell\,(M)) \\
(\mathrm{S}_\ell\mathrm{S}_{\mathcal{L}})\ \mathrm{S}_\ell\,(\mathrm{S}_{\mathcal{L}}\,(M,N), P) & (\mathrm{K}_\ell\mathrm{S}_{\mathcal{L}+1})\ \mathrm{K}_\ell\,(\mathrm{S}_{\mathcal{L}}\,(M,P)) \\
\quad \to \mathrm{S}_{\mathcal{L}-1}\,(\mathrm{S}_\ell\,(M,P), \mathrm{S}_\ell\,(N,P)) & \quad \to \mathrm{S}_{\mathcal{L}+1}\,(\mathrm{K}_\ell\,(M), \mathrm{K}_\ell\,(P))
\end{array}$$

Fig. 4. Labeled SKInT reduction rules (for every $0 \le \ell < \mathcal{L}$)

Lemma 3. *If $M \longrightarrow N$ in the labeled SKInT calculus, then for every $\boldsymbol{m} \in \mathcal{M}^\omega$, $[\![M]\!]_\omega\,(\boldsymbol{t}) \longrightarrow^* [\![N]\!]_\omega\,(\boldsymbol{t})$ in the λ_η^∞-calculus. Moreover, $[\![M]\!]_\omega\,(\boldsymbol{t}) \longrightarrow^+ [\![N]\!]_\omega\,(\boldsymbol{t})$ if $M \longrightarrow N$ by $(\mathrm{SI}_\ell^{\boldsymbol{m}})$ or (SK_ℓ).*

Proof. This is similar to the proof of Theorem 4.13 in [7], so let us proceed quickly. First, observe that if $\boldsymbol{t}$ and $\boldsymbol{t}'$ differ only at position i, and $t_i \longrightarrow^+ t_i'$ (resp. $\longrightarrow^*$), then $[\![P]\!]_\omega\,(\boldsymbol{t}) \longrightarrow^+ [\![P]\!]_\omega\,(\boldsymbol{t}')$ (resp. $\longrightarrow^*$) for every labeled SKInT term P: this is by structural induction on P. (This is where the use of $\oplus$ is crucial: it allows us to keep t_ℓ in the term $t_\ell \oplus t_{\ell+1}$.) This allows us to prove the Lemma by induction on the depth of the contracted redex in M, and handles the inductive case. The only remaining cases are the base cases, when M itself is the contracted redex. The case of rule $(\mathrm{SI}_\ell^{\boldsymbol{m}})$ has been dealt with above. The other rules are easy but boring computations. □

3.2 Local Strong Normalization of $\boldsymbol{\lambda_\eta^\infty}$

The purpose of this section is to show that by limiting nestings of underlinings in λ_η^∞-reduction to some fixed bound p, all reductions terminate. Equivalently, let

$(\mathcal{M}, 1, \bullet, m \mapsto \underline{m})$ be the label structure where $\mathcal{M}\hat{=}\mathbb{N}$ with its natural ordering, $\bullet$ is $\max$, the unit is 0, and $\underline{m}$ is defined by $\underline{m}\hat{=}m+1$. Adjoin a zero: this is traditionally written $+\infty$ (to be consistent with common usage, then, we let $\underline{+\infty} = +\infty+1 = +\infty$).

That is, we shall prove:

Theorem 1. *Fix $p \in \mathbb{N}$. On the labeled structure $(\mathbb{N} \cup \{+\infty\}, 0, \max, m \mapsto m+1)$, every λ^{∞}_{η}-reduction where $m < p$ in rule (β^{ω}) is finite.*

First three lemmas that hold in any labeled structure, with the general or the restricted (β^{ω}) rule.

Lemma 4. *If $s \longrightarrow t$ are terms then $(s)^m \longrightarrow (t)^m$.*

Proof. Easy structural induction on s. Observe that this works in the case of $\longrightarrow_p$ because (β^{ω})-redexes are *not* terms, but bodies. □

Lemma 5. *For every term s, $(s[\boldsymbol{x} := \boldsymbol{t}])^{\boldsymbol{m}} = (s)^{\boldsymbol{m}}[\boldsymbol{x} := \boldsymbol{t}]$.*

Lemma 6. *If $s[\boldsymbol{x} := \boldsymbol{t}] \longrightarrow^* \lambda^m \boldsymbol{y} \cdot e$ in λ^{∞}_{η}, then either:*

(i) *$s \longrightarrow^* \lambda^m \boldsymbol{y} \cdot e'$ for some body e' such that $e'[\boldsymbol{x} := \boldsymbol{t}] \longrightarrow^* e$, or*
(ii) *$s \longrightarrow^* x_i^{m_1}$ for some $i \in \mathbb{N}$, and $t_i \longrightarrow^* \lambda^{m_2} \boldsymbol{y} \cdot e$ with $m = m_2 \bullet m_1$.*

Proof. First observe that: (a) if $s' \oplus t' \longrightarrow^* \lambda^m \boldsymbol{y} \cdot e$ then $t' \longrightarrow^* \lambda^m \boldsymbol{y} \cdot e$ by a shorter rewrite sequence. This is an easy induction on the length of the given rewrite: the given rewrite cannot be empty, so consider the first redex. If the first reduction occurs in s' or in t', appeal to the induction hypothesis; if instead we reduce $s' \oplus t'$ by $(\oplus{-})$, this is clear; if $s' = s'' \oplus s'''$ and the first reduction is by $(\oplus)$, then by induction hypothesis there is an even shorter rewrite from $s''' \oplus t'$ to $\lambda^m \boldsymbol{y} \cdot e$, then from t' to $\lambda^m \boldsymbol{y} \cdot e$ by induction hypothesis again.

Then: (b) if $(s')^{m_1} \longrightarrow t''$ then $s' \longrightarrow t'$ for some t' such that $(t')^{m_1} = t''$. This is by structural induction on s'. Indeed, if this is by $(\oplus{-})$ at the top then $s' = s'' \oplus s'''$ and $t'' = (s''')^{m_1}$, so take $t'\hat{=}s'''$. The cases of $(\oplus)$ at the top, or where reduction does not occur at the top of terms are immediate.

Therefore: (c) if $(s')^{m_1} \longrightarrow^* t''$ then $s' \longrightarrow^* t'$ for some t' such that $(t')^{m_1} = t''$.

We now prove the Lemma by induction on the length k of the given reduction from $s[\boldsymbol{x} := \boldsymbol{t}]$ to $\lambda^m \boldsymbol{y} \cdot e$. If $k = 0$ then this is trivial; notice in particular that s cannot be of the form $u \oplus v$. So let $k \geq 1$. If s is of the form $u \oplus v$, so by (a) with $s'\hat{=}u[\boldsymbol{x} := \boldsymbol{t}]$ and $t'\hat{=}v[\boldsymbol{x} := \boldsymbol{t}]$, v rewrites to $\lambda^m \boldsymbol{y} \cdot e$ by a rewrite of length at most $k-1$, whence we conclude by induction. If s is of the form $\lambda^{m'} \boldsymbol{y} \cdot e'$ (with the same $\boldsymbol{y}$, using infinite α-renaming), then since all rewrites in $s[\boldsymbol{x} := \boldsymbol{t}] = \lambda^{m'} \boldsymbol{y} \cdot e'[\boldsymbol{x} := \boldsymbol{t}]$ occur in the body $e'[\boldsymbol{x} := \boldsymbol{t}]$, in particular $m' = m$ and *(i)* holds. If s is of the form z^{m_1} for some variable z, then first z must be some x_i, $i \in \mathbb{N}$—otherwise $s[\boldsymbol{x} := \boldsymbol{t}]$ cannot rewrite to $\lambda^m \boldsymbol{y} \cdot e$—and $s[\boldsymbol{x} := \boldsymbol{t}] = (t_i)^{m_1}$. By (c) with $s'\hat{=}t_i$ and $t'' = \lambda^m \boldsymbol{y} \cdot e$, $t_i \longrightarrow^* t'$ for some t' such that $(t')^{m_1} = \lambda^m \boldsymbol{y} \cdot e$; in particular $t' = \lambda^{m_2} \boldsymbol{y} \cdot e$ for some m_2 such that $m = m_2 \bullet m_1$, so *(ii)* holds. □

We now imitate van Daalen's proof of termination of the labeled λ-calculus [3]. Fix $p \in \mathbb{N}$, let $\longrightarrow_p$ be the reduction relation of λ^{∞}_{η} restricted to the case $m < p$ in rule (β^{ω}). Let SN_p be the set of all terms, bodies, or argument lists that are $\longrightarrow_p$-strongly normalizing.

Lemma 7. *For every body $e_0 \in SN_p$, for every argument list $\boldsymbol{t} \in SN_p$, $e_0[\boldsymbol{x} := \boldsymbol{t}]$ is in SN_p.*

Proof. Let $\succ_p$ be the strict ordering on $\mathbb{N}$ defined by $m \succ_p n$ iff $m < p$ and $m < n$. This is well-founded. Define $\mathrm{top}(\boldsymbol{t}) \hat{=} \min_{i \in \mathbb{N}} \mathrm{top}(t_i)$. We prove the lemma by induction on the pair $(\mathrm{top}(\boldsymbol{t}), e_0)$ ordered in the lexicographic product of $\succ_p$ on $\mathbb{N}$ and $\longrightarrow_p$ on SN_p—so this ordering is again well-founded.

Write e_0 as $s(\boldsymbol{t'})$. Notice that, because $\boldsymbol{t'}$ is of the form $t'_1, \ldots, t'_k, \boldsymbol{x'^{m'}}$, rewriting in $e[\boldsymbol{x} := \boldsymbol{t}]$ can only occur either at the top, or in finitely many subterms: $s[\boldsymbol{x} := \boldsymbol{t}]$, $t'_1[\boldsymbol{x} := \boldsymbol{t}]$, …, $t'_k[\boldsymbol{x} := \boldsymbol{t}]$. So any infinite rewrite starting from $e[\boldsymbol{x} := \boldsymbol{t}]$ must eventually rewrite at the top by contracting a (β^ω)-redex. More precisely, any infinite rewrite must rewrite $s[\boldsymbol{x} := \boldsymbol{t}]$ to some term $\lambda^m \boldsymbol{y} \cdot e$, each $t'_i[\boldsymbol{x} := \boldsymbol{t}]$ to some term t''_i, $1 \leq i \leq k$, and then contract the (β^ω)-redex $(\lambda^m \boldsymbol{y} \cdot e)(t''_1, \ldots, t''_k, \boldsymbol{x'^{m'}})$ to $e\left[\boldsymbol{y} := (t''_1, \ldots, t''_k, \boldsymbol{x'^{m'}})^{(m+1)^\omega}\right]$. Since the given rewrite is infinite: (a) $e\left[\boldsymbol{y} := (t''_1, \ldots, t''_k, \boldsymbol{x'^{m'}})^{(m+1)^\omega}\right]$ is not in SN_p. Also since the restricted form of (β^ω) applies: (b) $m < p$. By Lemma 6, either:

(i) $s \longrightarrow^* \lambda^m \boldsymbol{y} \cdot e'$ and: (c) $e'[\boldsymbol{x} := \boldsymbol{t}] \longrightarrow^* e$; then $e = s(\boldsymbol{t'}) \longrightarrow^* (\lambda^m \boldsymbol{y} \cdot e')(\boldsymbol{t'}) \longrightarrow e'\left[\boldsymbol{y} := (\boldsymbol{t'})^{(m+1)^\omega}\right]$. Since $\mathrm{top}\left((\boldsymbol{t'})^{(m+1)^\omega}\right) \geq m + 1$, and $m < p$ by (b), it obtains $\mathrm{top}\left((\boldsymbol{t'})^{(m+1)^\omega}\right) \prec_p m$. So the induction hypothesis applies: $e'\left[\boldsymbol{y} := (\boldsymbol{t'})^{(m+1)^\omega}\right] \in SN_p$. Since e_0 rewrites in at least one step to the latter body, i.e., the latter is less than e_0 in $\longrightarrow_p$, the induction hypothesis applies again, so $e'\left[\boldsymbol{y} := (\boldsymbol{t'})^{(m+1)^\omega}\right][\boldsymbol{x} := \boldsymbol{t}] \in SN_p$. But the latter is $(e'[\boldsymbol{x} := \boldsymbol{t}])\left[\boldsymbol{y} := (\boldsymbol{t'})^{(m+1)^\omega}[\boldsymbol{x} := \boldsymbol{t}]\right] = (e'[\boldsymbol{x} := \boldsymbol{t}])\left[\boldsymbol{y} := (\boldsymbol{t'}[\boldsymbol{x} := \boldsymbol{t}])^{(m+1)^\omega}\right]$ (by Lemma 5) $= (e'[\boldsymbol{x} := \boldsymbol{t}])\left[\boldsymbol{y} := \left((t'_1[\boldsymbol{x} := \boldsymbol{t}])^{m+1}, \ldots, (t'_k[\boldsymbol{x} := \boldsymbol{t}])^{m+1}, (\boldsymbol{x'^{m'}})^{(m+1)^\omega}\right)\right]$. This is in SN_p, and rewrites to $e\left[\boldsymbol{y} := \left((t''_1)^{m+1}, \ldots, (t''_k)^{m+1}, (\boldsymbol{x'^{m'}})^{(m+1)^\omega}\right)\right]$ (by (c), and using $t'_i[\boldsymbol{x} := \boldsymbol{t}] \longrightarrow^* t''_i$ together with Lemma 4), contradicting (a). Or:

(ii) $s \longrightarrow^* x_i^{m_1}$ for some $i \in \mathbb{N}$, and $t_i \longrightarrow^* \lambda^{m_2} \boldsymbol{y} \cdot e$ with $m = \max(m_2, m_1)$. Observing that top labels of terms are preserved during reduction, $\mathrm{top}(t_i) = m_2$, so $\mathrm{top}(\boldsymbol{t}) \leq m_2 \leq m$. But then $\mathrm{top}\left((t''_1, \ldots, t''_k, \boldsymbol{x'^{m'}})^{(m+1)^\omega}\right) \geq m + 1 > \mathrm{top}(\boldsymbol{t})$. Since $\mathrm{top}(\boldsymbol{t}) \leq m < p$ by (b), $\mathrm{top}(\boldsymbol{t}) \succ_p \mathrm{top}\left((t''_1, \ldots, t''_k, \boldsymbol{x'^{m'}})^{(m+1)^\omega}\right)$, so the induction hypothesis applies: $e\left[\boldsymbol{y} := (t''_1, \ldots, t''_k, \boldsymbol{x'^{m'}})^{(m+1)^\omega}\right]$ is in SN_p, contradicting (a). □

Since argument lists contain only finitely many terms that are not variables, by induction terms contain only finitely many redexes. In particular the tree of all possible rewrites from a given term is finitely-branching. By König's Lemma, therefore, any term t in SN_p has a longest rewrite: let $\nu(t)$ be its length.

Lemma 8. *If s and t are in SN_p, then so is $s \oplus t$.*

Proof. Easy induction on $(\nu(s), s, \nu(t))$ ordered in the lexicographic product of $>$ on $\mathbb{N}$, the subterm ordering $\succ$ on terms, and $>$ on $\mathbb{N}$ again. Notice that $s \longrightarrow t$ implies $\nu(s) > \nu(t)$. The only subtle case is when $s \oplus t$ rewrites by $(\oplus)$ at the top: then $s = s_1 \oplus s_2$. Since $\nu(s) \geq \nu(s_2)$ and $s \succ s_2$, the induction hypothesis applies, therefore $s_2 \oplus t \in SN_p$. Since $\nu(s) \geq \nu(s_1)$ and $s \succ s_1$, by the induction hypothesis again $s_1 \oplus (s_2 \oplus t) \in SN_p$. □

Lemma 9. *If t is in SN_p, then $(t)^m$ is in SN_p.*

Proof. By structural induction on the term t, using Lemma 8 in case t is headed by $\oplus$. □

Lemma 10. *If s is in SN_p and $\boldsymbol{t}$ is in SN_p, then $s(\boldsymbol{t})$ is in SN_p.*

Proof. Otherwise there is an infinite rewrite starting from $s(\boldsymbol{t})$. Since s and $\boldsymbol{t}$ are in SN_p and $\boldsymbol{t}$ contains only finitely many non-variable entries, it must be that $s \longrightarrow^* \lambda^m \boldsymbol{y} \cdot e$, $\boldsymbol{t} \longrightarrow^* \boldsymbol{t'}$ and $e\left[\boldsymbol{y} := (\boldsymbol{t'})^{(m+1)^\omega}\right]$ is not in SN_p. Since $\boldsymbol{t} \in SN_p$, $\boldsymbol{t'} \in SN_p$, so by Lemma 9 $(\boldsymbol{t'})^{(m+1)^\omega} \in SN_p$. Since s, and therefore also e is in SN_p, by Lemma 7 $e\left[\boldsymbol{y} := (\boldsymbol{t'})^{(m+1)^\omega}\right] \in SN_p$, a contradiction. □

Theorem 1 follows: every term, body or argument list a is in SN_p, by structural induction on a. If a is a labeled variable, this is obvious; this is an easy appeal to the induction hypothesis if a is a labeled λ-abstraction; this is by Lemma 8 if a is headed by $\oplus$, and by Lemma 10 if a is a body.

3.3 Local Strong Normalization for SKInT

The strong normalization result for λ_η^∞ lifts immediately to SKInT:

Theorem 2. *Fix $p \in \mathbb{N}$. On the labeled structure $(\mathbb{N}, 0, \max, m \mapsto m+1)$, every* SKInT*-reduction where $m_0 < p$ in rule $(\mathrm{SI}_\ell^{\boldsymbol{m}})$ is finite.*

Proof. By Lemma 3 (in particular, see (1)) and Theorem 1. □

Note that we do not need a zero (i.e., $+\infty$) here: we only needed it to work out the translation to the λ_η^∞-calculus.

4 Confluence

Theorem 2 can be used, much as in then λ-calculus, to give an alternative proof of confluence of SKInT. Confluence was proved by a method of parallel reductions in [7].

Lemma 11. *The labeled* SKInT*-calculus is locally confluent.*

Proof. The critical pairs are as for the unlabeled calculus. We show the two most important ones.

Between βI (i.e., $(\mathrm{SI}_{\mathcal{L}})$) and $(\mathrm{S}_\ell\mathrm{S}_{\mathcal{L}})$, $0 \leq \ell < \mathcal{L}$: $\mathrm{S}_\ell\left(\mathrm{S}_{\mathcal{L}}\left(\mathrm{I}^{\boldsymbol{m}}_{\mathcal{L}}, N\right), P\right)$ rewrites by βI to $\mathrm{S}_\ell\left({}^{1^{\mathcal{L}}, \boldsymbol{m}_{1..\infty} \bullet \underline{m_0}^{\omega}}(N), P\right)$, and by $(\mathrm{S}_\ell\mathrm{S}_{\mathcal{L}})$ to $\mathrm{S}_{\mathcal{L}-1}\left(\mathrm{S}_\ell\left(\mathrm{I}^{\boldsymbol{m}}_{\mathcal{L}}, P\right), \mathrm{S}_\ell\left(N, P\right)\right) \longrightarrow \mathrm{S}_{\mathcal{L}-1}\left(\mathrm{I}^{\boldsymbol{m}}_{\mathcal{L}-1}, \mathrm{S}_\ell\left(N, P\right)\right) \longrightarrow {}^{1^{\mathcal{L}-1}, \boldsymbol{m}_{1..\infty} \bullet \underline{m_0}^{\omega}}\left(\mathrm{S}_\ell\left(N, P\right)\right)$ (by βI) $= \mathrm{S}_\ell\left({}^{1^{\mathcal{L}}, \boldsymbol{m}_{1..\infty} \bullet \underline{m_0}^{\omega}}(N), {}^{1^{\omega}}(P)\right) = \mathrm{S}_\ell\left({}^{1^{\mathcal{L}}, \boldsymbol{m}_{1..\infty} \bullet \underline{m_0}^{\omega}}(N), P\right)$.

Between βI (i.e., $(\mathrm{SI}_{\mathcal{L}})$) and $(\mathrm{K}_\ell\mathrm{S}_{\mathcal{L}})$, $0 \leq \ell < \mathcal{L}$: $\mathrm{K}_\ell\left(\mathrm{S}_{\mathcal{L}}\left(\mathrm{I}^{\boldsymbol{m}}_{\mathcal{L}}, N\right)\right)$ rewrites by βI to $\mathrm{K}_\ell\left({}^{1^{\mathcal{L}}, \boldsymbol{m}_{1..\infty} \bullet \underline{m_0}^{\omega}}(N)\right)$, and by $(\mathrm{K}_\ell\mathrm{S}_{\mathcal{L}})$ to $\mathrm{S}_{\mathcal{L}+1}\left(\mathrm{K}_\ell\left(\mathrm{I}^{\boldsymbol{m}}_{\mathcal{L}}\right), \mathrm{K}_\ell\left(N\right)\right) \longrightarrow \mathrm{S}_{\mathcal{L}+1}\left(\mathrm{I}^{\boldsymbol{m}}_{\mathcal{L}+1}, \mathrm{K}_\ell\left(N\right)\right) \longrightarrow {}^{1^{\mathcal{L}+1}, \boldsymbol{m}_{1..\infty} \bullet \underline{m_0}^{\omega}}\left(\mathrm{K}_\ell\left(N\right)\right) = \mathrm{K}_\ell\left({}^{1^{\mathcal{L}}, \boldsymbol{m}_{1..\infty} \bullet \underline{m_0}^{\omega}}(N)\right)$. □

By Theorem 2, it follows that any subcalculus of labeled SKInT where the number of underlinings is bounded by some constant p is confluent. Now let M be any (unlabeled) SKInT term that reduces in m steps to t, in n steps to u. Decorate M with labels 0^{ω}, yielding a labeled SKInT term M'. Then M' rewrites to terms t' and u' such that, first, t and u are obtained by removing labels from t' and u' respectively; second, the labels in t' and u' are at most $p \hat{=} \max(m, n)$. We can then rewrite both t' and u' to some common term v' in labeled SKInT. Erase all labels, getting the unlabeled SKInT term v: then $t \longrightarrow^* v$ and $u \longrightarrow^* v$. So SKInT is confluent.

The same argument shows:

Theorem 3. *The labeled* SKInT *calculus is confluent.*

The labeled SKInT *calculus on the labeled structure* $(\mathbb{N}, 0, \max, m \mapsto m+1)$ *with rule* (SI_ℓ) *restricted to* $m_0 < p$*, for fixed* $p \in \mathbb{N}$*, is confluent and strongly normalizing.*

5 Labels in the λ-Calculus and Superdevelopments

Let the labeled λ-terms be defined in the standard way:

$$s, t ::= x | st | \lambda x \cdot t | (t)^m$$

The construction $(t)^m$ is now a term forming operation, not a meta-notation for an action on λ-terms.

Extend the $t \mapsto t^*$ translation of Figure 2 to *labeled* λ-terms as follows:

$$\begin{array}{rcl} x^* \hat{=} x^{1^{\omega}} & \qquad & [x]^m x^{\boldsymbol{m}} \hat{=} \mathrm{I}^{m,\boldsymbol{m}}_0 \\ (uv)^* \hat{=} \mathrm{S}_0\left(u^*, v^*\right) & & [x]^m y^{\boldsymbol{m}} \hat{=} \mathrm{K}_0\left(y^{\boldsymbol{m}}\right) \quad (y \neq x) \\ (\lambda x \cdot u)^* \hat{=} [x]^1 u^* & & [x]^m \mathrm{I}^{\boldsymbol{m}}_\ell \hat{=} \mathrm{I}^{\boldsymbol{m}}_{\ell+1} \\ ((u)^m)^* \hat{=} {}^{m^{\omega}}(u^*) & & [x]^m \mathrm{S}_\ell\left(M, N\right) \hat{=} \mathrm{S}_{\ell+1}\left([x]^m M, [x]^m N\right) \\ & & [x]^m \mathrm{K}_\ell\left(M\right) \hat{=} \mathrm{K}_{\ell+1}\left([x]^m M\right) \end{array}$$

Lemma 12. *For every labeled* SKInT *term* M, ${}^{\boldsymbol{m}'}([x]^m M) = [x]^{m'_0 \bullet m}\left(\boldsymbol{m}'_{1..\infty}(M)\right)$.

Proof. Easy structural induction on M.

Unlabeled SKInT has the property that $\mathrm{S}_0\left([x]M, N\right) \longrightarrow^+ M[x := N]$, which looks like saying that SKInT implements full call-by-name β-reduction. However this only implies that $\left((\lambda x \cdot u)v\right)^* = \mathrm{S}_0\left([x]u^*, v^*\right) \longrightarrow^+ u^*[x := v^*]$. The latter does not always rewrite to $(u[x := v])^*$, however it does so when v is a value in the sense of Plotkin. Similar properties hold here (define substitution in labeled SKInT so that $x^{\boldsymbol{m}}[x := N] \hat{=}^{\boldsymbol{m}}(N)$, $y^{\boldsymbol{m}}[x := N] \hat{=} y^{\boldsymbol{m}}$, the other clauses being obvious; in particular ${}^{\boldsymbol{m}}(M[x := N]) = {}^{\boldsymbol{m}}(M)[x := N]$):

Lemma 13. *For every labeled* SKInT *terms* M, N, $\mathrm{S}_0\left([x]^m M, N\right) \longrightarrow^+ M\left[x := {}^{\underline{m}^\omega}(N)\right]$.

Proof. By structural induction on M. If $M = x^{\boldsymbol{m}}$, $\mathrm{S}_0\left([x]^m M, N\right) = \mathrm{S}_0\left(\mathrm{I}_0^{m,\boldsymbol{m}}, N\right) \longrightarrow {}^{\boldsymbol{m}\bullet\underline{m}^\omega}(N) = M\left[x := {}^{\underline{m}^\omega}(N)\right]$. If $M = y^{\boldsymbol{m}}$ for some other variable y, $\mathrm{S}_0\left([x]^m M, N\right) = \mathrm{S}_0\left(\mathrm{K}_0\left(y^{\boldsymbol{m}}\right), N\right) \longrightarrow y^{\boldsymbol{m}} = M = M\left[x := {}^{\underline{m}^\omega}(N)\right]$. The other cases are straightforward. □

Define a labeled version of λ-reduction by the rule:

$$(\beta) \quad \left(\ldots((\lambda x \cdot u)^{m_1})^{m_2}\ldots\right)^{m_n} v \to \left(u\left[x := (v)^{\underline{m_n \bullet \ldots \bullet m_2 \bullet m_1}}\right]\right)^{m_n \bullet \ldots \bullet m_2 \bullet m_1}$$

where $m_n \bullet \ldots \bullet m_2 \bullet m_1$ denotes 1 if $n = 0$. Note again the difference with Hyland-Wadsworth-Lévy labeled reduction: the outer label is not underlined. Note also that Lemma 13 would suggest to drop the outer label entirely. This is deceiving: see the proof of Theorem 4 to understand where the outer label comes from.

The (β_V) rule of *call-by-value* λ-reduction in the sense of Plotkin is the restriction of (β) where V is a *value*:

$$V ::= x | \lambda x \cdot t | (V)^m$$

Lemma 14. *If* $M \longrightarrow N$ *in labeled* SKInT*, then* ${}^{\boldsymbol{m}}(M) \longrightarrow {}^{\boldsymbol{m}}(N)$.

Proof. Straightforward structural induction on M. □

Corresponding to values, SKInT-*values* are SKInT terms containing no subterm of the form $\mathrm{S}_0\left(M, N\right)$.

Lemma 15. *The following hold:*

1. *If* t *is a value, then* t^* *is a* SKInT*-value.*
2. *For every* SKInT*-value* W, $\mathrm{K}_0\left(W\right) \longrightarrow^* [y]^m W$ *for every variable* y *not free in* W *and every* $m \in \mathcal{M}$.
3. *For every distinct variables* x *and* y*, for every* SKInT *term* M *and every* SKInT*-value* W *not containing* y *free,* $\left([y]^m M\right)[x := W] \longrightarrow^* [y]^m(M[x := W])$.
4. *For every term* u *and value* V*, for every variable* x *not free in* V, $u^*[x := V^*] \longrightarrow^* (u[x := V])^*$.

Proof. Claims 1 and 2 are easy structural structural inductions on t and W respectively. Claim 3 is by structural induction on M. When M is a variable $z^{\boldsymbol{m}}$, by assumption $z \neq y$, so $[y]^m M[x := W] = \mathrm{K}_0\,(z^{\boldsymbol{m}})\,[x := W]$. If $z = x$, then this is $\mathrm{K}_0\,\left({}^{\boldsymbol{m}}(W)\right) \longrightarrow^* [y]^m\,\left({}^{\boldsymbol{m}}(W)\right)$ (by Claim 2) $= [y]^m(M[x := W])$; otherwise $[y]^m M[x := W] = [y]^m M = [y]^m(M[x := W])$. The other cases are straightforward.

Claim 4 is by structural induction on u, using Claim 3 when u is a λ-abstraction, and Lemma 14 when u is of the form $(u')^m$. □

Theorem 4. *If $s \longrightarrow t$ by labeled (β_V), then $s^* \longrightarrow^+ t^*$ in labeled* SKInT.

Proof. By structural induction on s. The only interesting case is when s is itself the redex $(\ldots((\lambda x \cdot u)^{m_1})^{m_2}\ldots)^{m_n} V$. Then, letting $m \hat{=} m_n \bullet \ldots \bullet m_2 \bullet m_1$, $s^* = \mathrm{S}_0\left({}^{m^\omega}([x]^1 u^*), V^*\right) = \mathrm{S}_0\left([x]^m\left({}^{m^\omega}(u^*)\right), V^*\right)$ (by Lemma 12) $\longrightarrow^+ {}^{m^\omega}(u^*)\left[x := {}^{\underline{m}^\omega}(V^*)\right]$ (by Lemma 13) $= {}^{m^\omega}\left(u^*\left[x := {}^{\underline{m}^\omega}(V^*)\right]\right) = {}^{m^\omega}\left(u^*\left[x := ((V)^{\underline{m}})^*\right]\right) \longrightarrow^* {}^{m^\omega}\left((u\,[x := (V)^{\underline{m}}])^*\right)$ (by Lemma 15, Claim 4, and Lemma 14) $= \left((u\,[x := (V)^{\underline{m}}])^m\right)^*$. □

In particular:

Theorem 5. *Fix $p \in \mathbb{N}$. On the labeled structure $(\mathbb{N}, 0, \max, m \mapsto m + 1)$, every labeled reduction where $m_1, \ldots, m_n < p$ in rule (β_V) is finite.*

Proof. By Theorem 2 and Theorem 4. □

The case $p = 1$, in particular, yields a call-by-value form of notion of *superdevelopments*, as introduced by Aczel in 1978 (see [10], Section 13.2), where not just residuals of initial redexes can be contracted, but also all redexes created upwards.

We do not know whether Theorem 5 holds also for general, call-by-name λ-reduction. Proof arguments such as given in [3], 14.1.10 and 14.1.11 in particular do not go through with this relaxed notion of labeled reduction. Neither does the L-translation of [7] or any other translation that we know of from call-by-name to call-by-value seem to help.

6 Conclusion

We have introduced a notion of labeled reduction for the SKInT-calculus, a first-order calculus that implements call-by-value λ-reduction naturally. This notion is more complex than in the λ-calculus, and is based on infinite lists of Lévy labels. This arises naturally from a translation to an infinitely η-expanded version of the λ-calculus, the λ^∞_η-calculus. While this may look like a trick, this should rather be taken as a way of lifting a yet undefined notion of paths [2] in the λ^∞_η-calculus to SKInT. The $\oplus$ operator of λ^∞_η, which was originally a mere technical device, appears in this context as an essential additional construct in defining call-by-value in path-based calculi. We plan to explore this issue in the future.

SKInT labels also allow us to reprove that SKInT is confluent, which was somewhat to be expected. However, Theorem 2 shows more: SKInT is *locally strongly normalizing*

(as the λ-calculus; we borrow the phrase from [3]), in the sense that any finite family of finite SKInT rewrites can be embedded in a confluent and terminating subcalculus, namely labeled SKInT where rule (SI_ℓ) is restricted to $m_0 < p$, for some fixed p.

Finally, we have shown that the labeled SKInT calculus implements correctly a liberalized version of the standard rule of labeled reduction, only for Plotkin's call-by-value reduction. This liberalization allows one to copy labels on the outside of terms without any underlining (without any consumption of resources), while still remaining locally strongly normalizing.

From another perspective, this modified labeled reduction rule defines a liberalized notion of redex family [11], where families are larger: redexes created upwards by some λ-abstraction are put in the same family as all initial redexes having the same λ-abstraction in functional position. We conjecture that sharing implementations of SKInT should provide optimal implementations, in a sense close to Lévy: all redexes in an extended family should be contracted at once.

References

1. M. Abadi, L. Cardelli, P.-L. Curien, and J.-J. Lévy. Explicit substitutions. In *Proceedings of the 17th Annual ACM Symposium on Principles of Programming Languages*, pages 31–46, San Francisco, California, January 1990.
2. A. Asperti and C. Laneve. Paths, computations and labels in the λ-calculus. In *RTA'93*, pages 152–167. Springer Verlag LNCS 690, 1993.
3. H. Barendregt. *The Lambda Calculus, Its Syntax and Semantics*, volume 103 of *Studies in Logic and the Foundations of Mathematics*. North-Holland Publishing Company, Amsterdam, 1984.
4. P. N. Benton, G. M. Bierman, and V. C. V. de Paiva. Computational types from a logical perspective I. *Journal of Functional Programming*, 8(2):177–193, 1998.
5. V. Danos. Personal communication, 2001.
6. R. David and B. Guillaume. A λ-calculus with explicit weakening and explicit substitution. *MSCS*, 11:169–206, 2001.
7. H. Goguen and J. Goubault-Larrecq. Sequent combinators: A Hilbert system for the lambda calculus. *MSCS*, 10(1):1–79, 2000.
8. J. Goubault-Larrecq. Conjunctive types and SKInT. In *Types'98*, pages 106–120. Springer Verlag LNCS 1657, 1999.
9. J. M. E. Hyland. A syntactic characterization of the equality in some models of the λ-calculus. *Journal of the London Mathematical Society*, 12(2):361–370, 1976.
10. J. W. Klop, V. van Oostrom, and F. van Raamsdonk. Combinatory reduction systems: Introduction and survey. *TCS*, 121(1–2):279–308, 1993.
11. J.-J. Lévy. *Réductions correctes et optimales dans le lambda calcul*. PhD thesis, Université Paris 7, 1978.
12. P.-A. Melliès. Typed lambda-calculi with explicit substitutions may not terminate. In *Proceedings of the CONFER workshop*, München, April 1994.
13. G. Plotkin. Call-by-name, call-by-value and the λ-calculus. *Theoretical Computer Science*, 1(2):125–159, 1975.
14. A. Ríos. *Contributions à l'étude des lambda-calculs avec substitutions explicites*. PhD thesis, École Normale Supérieure, December 1993.
15. C. P. Wadsworth. The relation between computational and denotation properties for Scott's D_∞-models of the lambda-calculus. *SIAM Journal of Computing*, 5:488–521, 1976.