INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET AUTOMATIQUE

A Few Remarks on SKInT

Jean Goubault-Larrecq

N ° 3475

Aout 1998

THÈME 2

Rapport de recherche

ISSN 0249-6399

A Few Remarks on SKInT

Jean Goubault-Larrecq*

Thème 2 — Génie logiciel
et calcul symbolique

Projet Coq

Rapport de recherche n° 3475 — Aout 1998

Abstract: SKIn and SKInT are two first-order languages that have been proposed recently by Healfdene Goguen and the author. While SKIn encodes lambda-calculus reduction faithfully, standardizes and is confluent even on open terms, it normalizes only weakly in the simply-typed case. On the other hand, SKInT normalizes strongly in the simply-typed case, standardizes and is confluent on open terms, and also encodes lambda-calculus reduction faithfully, although in a less direct way.
This report has two goals. First, we show that the natural simple type system for SKInT, seen as a natural deduction system, is not exactly a proof system for intuitionistic logic, but for a very close fragment of the modal logic S4, in which intuitionistic logic is easily coded. This explains why the SKIn and SKInT typing rules are different, and why SKInT encodes lambda-calculus in a less direct way than SKIn.
Second, we show that SKInT, like $\lambda\upsilon$ and a few other calculi of explicit substitutions, preserves strong normalization. In fact, SKInT also preserves weak normalization and solvability. We show this as a corollary of a stronger result, analogous to a well-known result in the lambda-calculus: the solvable SKInT-terms are exactly those that are typable in the system $S\omega$ of conjunctive types (inspired from Émilie Sayag), the weakly normalizing SKInT-terms (with or without η) are exactly those that have a definite positive $S\omega$-typing, and the strongly normalizing SKInT-terms (with or without η) are exactly those that are typable in the system S of conjunctive types, which does not have the universal type ω.

Key-words: conjunctive types, explicit substitutions, intuitionistic logic, lambda-calculus, normalization, S4 logic, sequent combinators, simple types, solvability, SKIn, SKInT.

(Résumé : tsvp)

This work has been done in the context of Dyade (R&D joint venture between Bull and Inria).

* Jean.Goubault@{dyade,inria}.fr

Unité de recherche INRIA Rocquencourt
Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
Téléphone : (33 1) 39 63 55 11 – Télécopie : (33 1) 39 63 53 30

Quelques remarques sur SKInT

Résumé : SKIn et SKInT sont deux langages du premier ordre proposés récemment par Healfdene Goguen et l'auteur. Si SKIn permet de modéliser la réduction du lambda-calcul fidèlement, standardise et est confluent même sur les termes ouverts, par contre il ne termine que faiblement dans le cas simplement typé. En revanche, SKInT termine fortement dans le cas simplement typé, standardise et est confluent sur les termes ouverts, et modélise la réduction du lambda-calcul fidèlement quoique d'une façon moins directe.

Ce rapport a deux buts. Premièrement, nous montrons que le système de types simples naturel pour SKInT, vu en tant que système de déduction naturelle, n'est pas exactement un système de preuve pour la logique intuitionniste, mais pour un fragment de la logique modale S4 qui en est très proche et dans lequel la logique intuitionniste est facilement codable. Ceci explique la différence des systèmes de typage entre SKIn et SKInT, ainsi que la raison pour laquelle SKInT modélise le lambda-calcul d'une façon moins directe que SKInT. Deuxièmement, nous montrons que SKInT, tout comme $\lambda\upsilon$ et quelques autres calculs de substitutions explicites, préserve la normalisation forte. En fait, SKInT préserve aussi la normalisation faible et la résolubilité, ce que nous montrons comme un corollaire d'un résultat plus fort, qui est l'analogue d'un résultat connu en lambda-calcul : les SKInT-termes résolubles sont exactement ceux qui sont typables dans le système de types conjonctifs $S\omega$ (inspiré des travaux d'Émilie Sayag), les SKInT-termes faiblement normalisants (avec ou sans η) sont exactement ceux qui ont un typage défini positif en $S\omega$, et les SKInT-terms fortement normalisants (avec ou sans η) sont exactement ceux qui sont typables dans le systèmes de types conjonctifs S, qui n'a pas le type universel ω.

Mots-clé : combinateurs séquentiels, lambda-calcul, logique intuitionniste, logique S4, normalisation, résolubilité, SKIn, SKInT, substitutions explicites, types conjonctifs, types simples.

Introduction

The λ-calculus and its typed versions are important tools for defining and studying most fundamental computation and deduction paradigms. However, the non-trivial nature of substitution, as used in the definition of λ-reduction notably, has spurred the design of various first-order languages meant to represent λ-terms, λ-reduction and λ-conversion, where computation is simple, first-order rewriting, and substitution becomes an easy notion again. Let us cite Curry's combinators [CF58], Curien's categorical combinators [Cur86], the myriad of so-called λ-calculi with explicit substitutions, among which $\lambda\sigma$ [ACCL90], $\lambda\sigma_{\Uparrow}$ [HL89], $\lambda\upsilon$ [LRD94], $\lambda\phi$ [MH96], etc. Unfortunately, each one of these calculi has defects: Curry's combinators do not modelize λ-conversion fully, categorical combinators, $\lambda\sigma$ and $\lambda\sigma_{\Uparrow}$ do not normalize strongly in the typed case [Mel95], $\lambda\upsilon$ is not confluent in the presence of free variables (i.e., meta-variables in explicit substitutions jargon), $\lambda\phi$ modelizes λ-conversion but not λ-reduction, etc.

SKInT [GGL98] is a first-order language and rewrite system that does not have these defects. It modelizes reduction in the λ-calculus, in the sense that there is a mapping L^* from λ-terms to SKInT such that whenever u rewrites to v in the λ-calculus, then $L^*(u)$ rewrites to $L^*(v)$ in SKInT. SKInT is confluent even on open terms, i.e. terms with meta-variables; L^* defines a convervative embedding of the λ-calculus inside SKInT, in that for any λ-terms u and v, u and v are λ-convertible if and only if $L^*(u)$ and $L^*(v)$ are convertible in SKInT; reduction in SKInT standardizes; L^* preserves weak normalization, i.e., if u is a weakly normalizing λ-term, then $L^*(u)$ is a weakly normalizing SKInT-term. SKInT also enjoys a simple type discipline corresponding to that of the λ-calculus, that is, if u is a λ-term of type τ, then $L^*(u)$ is a SKInT-term of some type $L^*(\tau)$ easily computed from τ; reduction in SKInT obeys subject reduction, and every simply-typed SKInT-term normalizes strongly.

Of course, SKInT has its own set of defects: first, it is an infinite first-order language, since it is built on an infinite signature, and reduction is defined by rule schemata, not just rules. Second, L^* is actually a composition $(t \mapsto t^*) \circ L$, where the map $t \mapsto t^*$ is itself a translation from λ-terms to SKInT, but one which does not preserve general λ-reduction, only some superset of call-by-value λ-reduction [Plo75]; L is a translation of the λ-calculus into the call-by-value λ-calculus. But then, we can replace L by any other such translation. That is, although SKInT can be viewed as a first-order implementation of the λ-calculus, it probably cannot be viewed so in a *natural* way.

The aim of this paper is, first, to address the latter problem and to show that the natural translation $t \mapsto t^*$ defines an isomorphism between SKInT and a calculus stricly between the call-by-value λ-calculus and the full λ-calculus; we call it λ_{clos}, and show that λ_{clos} is a language of proof-terms for a logic that is very close to intuitionistic logic: its Kripke models are just preorders, a.k.a. S4 frames, *without* the intuitionistic proviso that any propositional variable true in some world must remain true in all later worlds. We then show that reduction in λ_{clos}, and therefore SKInT, implements cut elimination for this logic.

Our second aim, then, is to extend our result on preservation of weak normalization to show that SKInT also preserves strong normalization—just like $\lambda\upsilon$—and solvability; this is done by showing that, just like in the λ-calculus, strongly normalizing, weakly normalizing and solvable terms are characterized as terms that are typable in various conjunctive type disciplines [CC90].

The plan of the paper is as follows: we introduce the required notions and notations in Section 1, then we attack our first goal in Section 2: call *near-intuitionistic logic* the logic whose Kripke semantics is just as implicational intuitionistic logic, except that true variables need not remain true in all later worlds; then we show that the typing rules for a variant of the λ-calculus that we call λ_{clos} is a sound and complete natural deduction system for near-intuitionistic logic (Theorem 1), that λ_{clos}-reduction implements cut-elimination for this logic (Theorem 2), and that λ_{clos} and SKInT are tightly related—in particular, that the typing rules of SKInT are sound and complete for near-intuitionistic logic, and that SKInT-reduction implements some form of cut-elimination (Theorem 3). In Section 3, we turn to our second goal, and show that any reasonable translation from the λ-calculus to SKInT preserves solvability, weak normalization and strong normalization (Corollary 6), where "reasonable" means that it preserves typability in certain systems of conjunctive types; this holds in particular for the translations of [GGL98]. In turn, this result follows from a result stating that all terms that are typable in particular systems of conjunctive types (to be defined in Section 3) have the corresponding normalization properties (Theorem 5). The converse also holds (Corollary 12), just as in the λ-calculus.

$$(SI_\ell) \quad S_\ell(I_\ell, w) \to w \qquad (SK_\ell) \quad S_\ell(K_\ell(u), w) \to u$$

$$\begin{array}{ll} (S_\ell I_\mathcal{L}) & S_\ell(I_\mathcal{L} w, \to) I_{\mathcal{L}-1} \\ (S_\ell K_\mathcal{L}) & S_\ell(K_\mathcal{L}(u), w) \to K_{\mathcal{L}-1}(S_\ell(u, w)) \\ (S_\ell S_\mathcal{L}) & S_\ell(S_\mathcal{L}(u, v), w) \\ & \qquad \to S_{\mathcal{L}-1}(S_\ell(u, w), S_\ell(v, w)) \end{array} \qquad \begin{array}{ll} (K_\ell I_\mathcal{L}) & K_\ell(I_{\mathcal{L}-1}) \to I_\mathcal{L} \\ (K_\ell K_\mathcal{L}) & K_\ell(K_{\mathcal{L}-1}(u)) \to K_\mathcal{L}(K_\ell(u)) \\ (K_\ell S_{\mathcal{L}+1}) & K_\ell(S_\mathcal{L}(u, w)) \to S_{\mathcal{L}+1}(K_\ell(u), K_\ell(w)) \end{array}$$

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

1 SKInT and the λ-Calculus

Recall that the syntax of the λ-calculus is [Bar84]:

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

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, ..., and variables by x, y, z, etc. α-equivalence is the compatible closure of:

$$(\alpha) \quad \lambda x \cdot (t[x/y]) = \lambda y \cdot t$$

and $t[s/x]$ denotes the usual capture-avoiding substitution of s for x in t. 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:

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

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.

We shall also add the following η-reduction rule:

$$(\eta) \quad \lambda x \cdot tx \to x \quad (x \text{ not free in } t)$$

to the λ-calculus, yielding the so-called λ_η-calculus. The corresponding compatible closure relation will sometimes be noted $\longrightarrow_\eta$ to distinguish it from $\longrightarrow$, and similarly for the calculi to come and their respective η rules.

The terms of SKInT, and of its companion calculus SKIn [GGL98], on the other hand, are defined by the grammar:

$$u ::= x \mid I_\ell \mid S_\ell(u, u) \mid K_\ell(u)$$

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.

SKIn is defined as SKInT, except that rule $(K_\ell S_{\mathcal{L}+1})$ is replaced by $(K_\ell S_\mathcal{L})$: $K_\ell(S_{\mathcal{L}-1}(u, v)) \to S_\mathcal{L}(K_\ell(u), K_\ell(v))$; conversely, SKInT is as SKIn, except that rule $(K_\ell S_\mathcal{L})$ is restricted to the case $\ell < \mathcal{L} - 1$.

Both SKIn and SKInT are confluent and standardize. The standard reductions are defined as follows. For every SKInT-term u, we can view u as a tree labelled by operators S_ℓ and K_ℓ, $\ell \geq 0$, and with leaves the constants I_ℓ and variables x. The *spine* of u is its leftmost branch. A *spine-reduction* is a sequence of rewrite steps where all contracted redexes are on the spine; we write $u \longrightarrow^s v$ when u rewrites to v by one spine-reduction. Spine-reductions play the role of head-reductions in the λ-calculus.

Every SKInT-term u can now be written as another tree, whose root is the spine S of u—more precisely, the leaf and the sequence of operators along the spine—and whose successors are the subterms $u_1, \ldots, u_n$ hanging off the spine—which we call the *arguments* of u. In this case, we also write u as $S[u_1, \ldots, u_n]$. For instance, consider $u =_{\mathrm{df}} S_3(K_1(S_2(I_0, S_1(x, I_1))), I_2)$. Its spine is the word $S =_{\mathrm{df}} S_3 K_1 S_2 I_0$, and we

$$
\begin{array}{lll}
 & & [x]x = I_0 \\
x^* & = x & [x]y = K_0(y) \quad (y \neq x) \\
(st)^* & = S_0(s^*, t^*) & [x](I_\ell) = I_{\ell+1} \\
(\lambda x \cdot t)^* & = [x](t^*) & [x](S_\ell(u,v)) = S_{\ell+1}([x]u, [x]v) \\
 & & [x](K_\ell(u)) = K_{\ell+1}([x]u)
\end{array}
$$

Figure 2: Translation from the λ-calculus to SKIn

can write u as $S[S_1(x, I_1), I_2]$. We can continue recursively, and write u as $S[S_1 x[I_1[]], I_2[]]$. We call such a notation a *tree of spines*.

A *standard reduction* $u \longrightarrow^{std*} v$ starts with 0 or more spine reductions from u to some term w, followed by standard reductions inside the arguments of w leading to v. This is well-defined, by induction on v viewed as a tree of spines. The standardization theorem [GGL98] states that whenever $u \longrightarrow^* v$ in SKInT, resp. SKIn, then $u \longrightarrow^{std*} v$.

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 [GGL98]).

We shall also consider SKInT$_\eta$, which is SKInT plus the following group η:

$$(\eta S_\ell) \quad S_{\ell+1}(K_\ell(u), I_\ell) \to u \quad (\ell \geq 0)$$

SKInT$_\eta$ is also confluent, and η-reductions can be postponed after all other reductions, just like in the λ-calculus.

The connection between the λ-calculus and SKInT, resp. SKIn, is given by natural translation functions going from one calculus to the other. First, we translate λ-terms to SKInT-terms, resp. SKIn-terms, by using the function $t \mapsto t^*$ defined in Figure 2.

The semantical idea behind SKInT will be made clear by stating the converse translation from SKInT to the λ-calculus. Intuitively, we have:

$$
\begin{array}{rcl}
I_\ell & \sim & \lambda x_0 \cdot \ldots \cdot \lambda x_{\ell-1} \cdot \lambda x_\ell \cdot x_\ell \\
S_\ell(u,v) & \sim & \lambda x_0 \cdot \ldots \cdot \lambda x_{\ell-1} \cdot u x_0 \ldots x_{\ell-1}(v x_0 \ldots x_{\ell-1}) \\
K_\ell(u) & \sim & \lambda x_0 \cdot \ldots \cdot \lambda x_{\ell-1} \cdot \lambda x_\ell \cdot u x_0 \ldots x_{\ell-1}
\end{array}
$$

It should be apparent that I_ℓ, S_ℓ, K_ℓ generalize Curry's combinators I, S and K respectively.

Formally, it is more interesting to define a more complicated translation $u \mapsto [\![u]\!]()$, shown in Figure 3. This translation in fact maps any SKInT-term u and any list $(s_0, \ldots, s_{n-1})$ of λ-terms to a λ-term $[\![u]\!](s_0, \ldots, s_{n-1})$. Then $u \mapsto [\![u]\!]()$ preserves reductions, i.e., if $u \longrightarrow^* v$ in SKInT, then $[\![u]\!]() \longrightarrow^*_\eta [\![v]\!]()$ in the λ_η-calculus. (In fact $[\![u]\!](s_0, \ldots, s_{n-1}) \longrightarrow^*_\eta [\![v]\!](s_0, \ldots, s_{n-1})$ for any $n \geq 0$ and any λ-terms s_0, …, s_{n-1}; we can replace $\longrightarrow_\eta$ by $\longrightarrow$ in the case of so-called well-staged terms u.)

Conversely, the $t \mapsto t^*$ translation also preserves reduction, in the following senses. First, if $s \longrightarrow^* t$ in the λ-calculus, then $s^* \longrightarrow^* t^*$ in SKIn (not SKInT); but if $s \longrightarrow^* t$ in the call-by-value λ-calculus, then $s^* \longrightarrow^* t^*$ in SKInT. Therefore, if L is a map from λ-terms to λ-terms such that $s \longrightarrow^* t$ in the λ-calculus entails $L(s) \longrightarrow^* L(t)$ in the call-by-value λ-calculus, then, defining L^* as the map sending t to $(L(t))^*$, it follows that $s \longrightarrow^* t$ in the λ-calculus implies $L^*(s) \longrightarrow^* L^*(t)$ in SKInT: see [GGL98].

Moreover, $t \mapsto t^*$ and $u \mapsto [\![u]\!]()$ are almost inverses of each other since for every λ-term t, $t \longrightarrow^* [\![t^*]\!]()$ in the λ-calculus. However there is in general no connection between a SKInT-term u and $([\![u]\!]())^*$: for instance, let u be $K_0(S_0(x,y))$, then $([\![u]\!]())^* = (\lambda x_0 \cdot xy)^* = S_1(K_0(x), K_0(y))$ is not convertible with u in SKInT—while it would be in SKIn.

$$
\begin{array}{rcl}
[\![x]\!](s_0,\ldots,s_{n-1}) & =_{\mathrm{df}} & x s_0 \ldots s_{n-1} \\
[\![I_\ell]\!](s_0,\ldots,s_{n-1}) & =_{\mathrm{df}} & \begin{cases} s_\ell s_{\ell+1}\ldots s_{n-1} & n > \ell \\ \lambda x_n \cdot \ldots \cdot \lambda x_\ell \cdot x_\ell & n \le \ell \end{cases} \\
[\![K_\ell(u)]\!](s_0,\ldots,s_{n-1}) & =_{\mathrm{df}} & \begin{cases} [\![u]\!](s_0,\ldots,s_{\ell-1},s_{\ell+1},\ldots,s_{n-1}) & n > \ell \\ \lambda x_n \cdot \ldots \cdot \lambda x_\ell \cdot [\![u]\!](s_0,\ldots,s_{n-1},x_n,\ldots,x_{\ell-1}) & n \le \ell \end{cases} \\
[\![S_\ell(u,v)]\!](s_0,\ldots,s_{n-1}) & =_{\mathrm{df}} & \begin{cases} [\![u]\!](s_0,\ldots,s_{\ell-1}, \\ \quad [\![v]\!](s_0,\ldots,s_{\ell-1}),s_\ell,\ldots,s_{n-1}) & n \ge \ell \\ \lambda x_n \cdot \ldots \cdot \lambda x_{\ell-1} \cdot \\ \quad [\![u]\!](s_0,\ldots,s_{n-1},x_n,\ldots,x_{\ell-1}, \\ \qquad [\![v]\!](s_0,\ldots,s_{n-1},x_n,\ldots,x_{\ell-1})) & n < \ell \end{cases}
\end{array}
$$

Figure 3: Interpretation of SKInT-terms as λ-terms

$$
\frac{}{\Gamma, x:\tau \vdash x:\tau} \qquad \frac{}{\Gamma \vdash I_\ell : \tau_0 \to \ldots \to \tau_{\ell-1} \to \tau_\ell \to \tau_\ell}
$$

$$
\frac{\Gamma \vdash u : \tau_0 \to \ldots \to \tau_{\ell-1} \to \tau_\ell \to \tau \quad \Gamma \vdash v : \tau_0 \to \ldots \to \tau_{\ell-1} \to \tau_\ell}{\Gamma \vdash S_\ell(u,v) : \tau_0 \to \ldots \to \tau_{\ell-1} \to \tau} \qquad \frac{\Gamma \vdash u : \tau_0 \to \ldots \to \tau_{\ell-1} \to \tau_\ell \to \tau}{\Gamma \vdash K_\ell(u) : \tau_0 \to \ldots \to \tau_{\ell-1} \to \tau' \to \tau_\ell \to \tau}
$$

Figure 4: Simple types for SKInT

The simple type discipline for the λ-calculus is defined by judgments $\Gamma \vdash t : \tau$, where t is a λ-term, τ is a simple type, i.e. a term in the following language:

$$\tau ::= B \mid \tau \to \tau$$

where B is a given non-empty set of so-called base types. Finally, Γ is a context, namely a finite map from variables to types; $\Gamma, x : \tau$ denotes Γ enriched by mapping x to τ, where x is outside the domain of Γ.

The typing rules for the λ-calculus are:

$$\frac{}{\Gamma, x:\tau \vdash x:\tau}$$

$$
\frac{\Gamma \vdash s : \tau_1 \to \tau_2 \quad \Gamma \vdash t : \tau_1}{\Gamma \vdash st : \tau_2} \qquad \frac{\Gamma, x : \tau_1 \vdash s : \tau_2}{\Gamma \vdash \lambda x \cdot s : \tau_1 \to \tau_2}
$$

On the other hand, the typing rules of SKInT are shown in Figure 4.

Both the λ-calculus and SKInT enjoy subject reduction and normalize strongly on simply-typed terms. Moreover, we have the following meta-theorem on SKInT. Call a context Γ *arrowed* if and only if Γ maps every variable in its domain to an arrow type (of the form $\tau_1 \to \tau_2$); then for every arrowed context Γ, if $\Gamma, x : \tau_1 \vdash u : \tau_2$, then $\Gamma \vdash [x]u : \tau_1 \to \tau_2$. The restriction to arrowed contexts is necessary: consider the case where u is a variable other than x.

2 SKInT, the λ_{clos}-Calculus and Near-Intuitionistic Logic

The latter meta-theorem suggests that SKInT would be closer to a form of λ-calculus where the typing rule of abstractions $\lambda x \cdot s$ would be as for the λ-calculus (see above) but only for Γ arrowed. We then get a variant of the modal logic S4, where arrows are thought of as modal boxes. We shall show that SKInT is indeed a sound and complete proof-term language for *near-intuitionistic logic*, namely the logic whose semantics is defined as follows:

Definition 1 *A* near-intuitionistic, or S4 frame *is a triple* $(\mathcal{W}, \leq, \rho)$*, where* $\mathcal{W}$ *is a non-empty set of worlds* w*,* $\leq$ *is a pre-order on* $\mathcal{W}$ *(a reflexive and transitive relation), and* ρ *is a valuation mapping base types to subsets of* $\mathcal{W}$*.*

The relation $\models_{(\mathcal{W},\leq,\rho)}$*, or* $\models$ *when the frame is clear, is defined by:*

$$\begin{array}{rcll} w \models b & \textit{iff} & w \in \rho(b) & \textit{(b a base type)} \\ w \models \tau_1 \to \tau_2 & \textit{iff} & \forall w' \cdot w \leq w' \Rightarrow w' \models \tau_1 \Rightarrow w' \models \tau_2 & \end{array}$$

Frames are defined as for the modal logic S4; the only difference with intuitionistic frames is that we do not require $\rho(b)$ to be upwards-closed, i.e. to be such that $w \in \rho(b)$ implies $w' \in \rho(b)$ for every w' such that $w \leq w'$. So Definition 1 can be seen as the semantics of a particular fragment of S4, where $\tau_1 \to \tau_2$ means the same as the S4 formula $\Box(\tau_1 \Rightarrow \tau_2)$.

We could certainly prove this directly on SKInT, but it will turn out to be profitable to go through another language of proof-terms, λ_{clos}, and to show, first, that the latter is a language of proof-terms for near-intuitionistic logic, and second, that it is isomorphic to SKInT(modulo convertibility, and in the presence of η rules). As λ_{clos} is very close to the λ-calculus, this will weave a more precise web of relationships between these languages. This will also show that, in a certain sense, SKInT-reduction, as λ_{clos}-reduction, represents cut-elimination in a sequent system for near-intuitionistic logic.

The λ_{clos}-*calculus* is inspired by the variant of Bierman-de Paiva's λ_{S4} [BdP92] used in [GL96]. The λ_{clos}-terms s, t, …, are defined as:

$$t ::= x \mid tt \mid \langle x \cdot t, \theta \rangle$$

where θ is an explicit *substitution*, i.e. a finite mapping from variables to λ_{clos}-terms. Terms like $\langle x \cdot t, \theta \rangle$ are called *closures*, $x \cdot t$ is the *code* part of the closure, x is its *argument*, t is its *body* and θ is the *environment* part; intuitively, closures are pairs of a piece of code computing the value of t when given a value for x as argument in the environment θ; the role of θ is to map the free variables of t (except x) to their respective values while evaluating t. We also write $\{x_1 := t_1, \ldots, x_k := t_k\}$ for the explicit substitution mapping x_i to t_i, $1 \leq i \leq k$.

The variable x and the variables in the domain of θ are bound. Moreover, we constrain λ_{clos}-closures $\langle x \cdot t, \theta \rangle$ to be such that every free variable of t except x is in the domain of θ. Substitution is defined as usual, but the latter constraint entails that substitution on closures only operates on the environment, not the body: $\langle x \cdot t, \{x_1 := t_1, \ldots, x_k := t_k\}\rangle\sigma = \langle x \cdot t, \{x_1 := t_1\sigma, \ldots, x_k := t_k\sigma\}\rangle$, where σ is an (implicit, ordinary) substitution.

We adopt Barendregt's convention on variable naming, so that no variable is bound at two distinct occurrences or occurs both bound and free in a term; we shall sometimes violate this condition (like in the definition of $\langle x \cdot t\rangle$ below) in the name of increased readability, under the convention that the intended term is obtained by some obvious α-renaming. Finally, α-equivalent terms are equated, where α-equivalence is the smallest congruence $=$ such that:

$$\begin{array}{ll} (\alpha) & \langle x' \cdot t[x'/x, x'_1/x_1, \ldots, x'_k/x_k], \{x'_1 := t_1, \ldots, x'_k := t_k\}\rangle \\ & = \langle x \cdot t, \{x_1 := t_1, \ldots, x_k := t_k\}\rangle \end{array}$$

where x, x_1, …, x_k are pairwise distinct, and so are x', x'_1, …, x'_k.

To make the notation somewhat more transparent, we define $\langle x \cdot t\rangle$ as an abbreviation for $\langle x \cdot t, \{x_1 := x_1, \ldots, x_k := x_k\}\rangle$, where x_1, …, x_k are the free variables of t except x. We call terms $\langle x \cdot t\rangle$ *abstractions*; it follows that every closure $\langle x \cdot t, \theta\rangle$ can be written as an abstraction on which some (implicit, ordinary) substitution $\hat{\theta}$ has been applied: if θ is $\{x_1 := t_1, \ldots, x_k := t_k\}$, $\hat{\theta}$ is the substitution $[t_1/x_1, \ldots, t_k/x_k]$, then $\langle x \cdot t, \theta\rangle = \langle x \cdot t\rangle\hat{\theta}$.

$$\frac{}{\Gamma, x : \tau \vdash x : \tau}$$

$$\frac{\Gamma \vdash s : \tau_1 \to \tau_2 \quad \Gamma \vdash t : \tau_1}{\Gamma \vdash st : \tau_2} \qquad \frac{\begin{array}{c}\Gamma \vdash t_1 : \tau_1' \to \tau_1'' \quad \dots \quad \Gamma \vdash t_k : \tau_k' \to \tau_k'' \\ x_1 : \tau_1' \to \tau_1'', \dots, x_k : \tau_k' \to \tau_k'', x : \tau' \vdash s : \tau''\end{array}}{\Gamma \vdash \langle x \cdot s, \{x_1 := t_1, \dots, x_k := t_k\}\rangle : \tau' \to \tau''}$$

Figure 5: Typing λ_{clos}

The reduction relation of λ_{clos} is defined as the compatible closure of:

$$\begin{array}{lll}
(\beta) & \langle x \cdot t, \theta\rangle s & \to \quad t(\hat\theta \cup [s/x]) \\
(\iota) & \multicolumn{2}{l}{\langle x \cdot t, \{x_1 := \langle y \cdot s, \{y_1 := s_1, \dots, y_p := s_p\}\rangle, x_2 := t_2, \dots, x_k := t_k\}\rangle} \\
& & \to \quad \langle x \cdot t[\langle y \cdot s, \{y_1 := y_1, \dots, y_p := y_p\}\rangle / x_1] \\
& & \qquad \{y_1 := s_1, \dots, y_p := s_p, x_2 := t_2, \dots, x_k := t_k\}\rangle \\
(0) & \multicolumn{2}{l}{\langle x \cdot t, \{x_1 := t_1, x_2 := t_2, \dots, x_k := t_k\}\rangle} \\
& & \to \quad \langle x \cdot t, \{x_2 := t_2, \dots, x_k := t_k\}\rangle \qquad (x_1 \text{ not free in } t) \\
(2) & \multicolumn{2}{l}{\langle x \cdot t, \{x_1 := t_1, x_2 := t_2, \dots, x_k := t_k\}\rangle} \\
& & \to \quad \langle x \cdot t[x_2/x_1], \{x_2 := t_2, \dots, x_k := t_k\}\rangle \qquad (\text{if } t_1 = t_2)
\end{array}$$

Some comments are in order. Rule (β) is essentially just ordinary β-reduction for closures; $\hat\theta \cup [s/x]$ is the (implicit) substitution mapping x to s and every other variable to its image under $\hat\theta$. Because of the variable naming convention, it would be equivalent to write $t\hat\theta[s/x]$ or $t[s/x]\hat\theta$ instead of $t(\hat\theta \cup [s/x])$.

The idea of rule (ι) is that whenever some free variable x_1 in t (except x) is mapped to a closure $\langle y \cdot s, \{y_1 := s_1, \dots, y_p := s_p\}\rangle$, we can push the code part $y \cdot s$ of x_1 inside the body t of the closure itself, retaining the environment part $y_1 := s_1, \dots, y_p := s_p$ in the environment of the outer closure; this can be thought as an inlining rule, as used in compilers for functional languages. Note that we might have chosen any x_i instead of x_1, but since environments are functions, bindings $x_i := t_i$ commute freely: choosing x_1 does not entail any loss of generality. Logicians will also notice from the typing rules to come that this is the box-under-box rule of S4 or of linear logic.

Rule (0) is a garbage collection rule: it expresses that there is no need to keep a binding $x_1 := t_1$ in the environment when the code part does not refer to x_1. Rule (2) is a contraction rule: if x_1 and x_2 are bound to the same term t_1 in the environment, we can eliminate one binding, say $x_1 := t_1$, and replace x_1 by x_2 in the body t of the code part.

Note by the way that λ_{clos} naturally defines a superset of the call-by-value λ-calculus [Plo75]. Indeed, translate call-by-value λ-terms by: $f(x) =_{\mathrm{df}} x$, $f(st) =_{\mathrm{df}} f(s)f(t)$, and $f(\lambda x \cdot s) =_{\mathrm{df}} \langle x \cdot f(s)\rangle$. Then, for every value V (a value being any term except an application), $f((\lambda x \cdot s)V)$ rewrites to $f(s[V/x])$ in λ_{clos}: indeed, it is clear that the former rewrites to $f(s)[f(V)/x]$, and it remains to show that the latter rewrites to $f(s[V/x])$. This is by structural induction on s, and the only non-trivial case is when s is a λ-abstraction $\lambda y \cdot t$: then $f(s)[f(V)/x] = \langle y \cdot f(t)\rangle[f(V)/x]$; if V is a variable, then the result follows by α-renaming, and if V is a λ-abstraction, then $f(V)$ is a closure, and the result follows by rule (ι) and the induction hypothesis.

We shall also consider the following η-rule, yielding the $\lambda_{\mathrm{clos}\,\eta}$-calculus:

$$(\eta) \quad \langle x \cdot zx, \theta\rangle \to z\hat\theta \qquad (z \neq x)$$

Notice that z here is a *variable*, not any term with x not free in it, as in the λ-calculus.

The simple typing rules for λ_{clos} are as shown in Figure 5.

The typing rule for closures can be viewed as a combination of a rule to type abstractions:

$$\frac{\Gamma, x : \tau' \vdash s : \tau''}{\Gamma, \Delta \vdash \langle x \cdot s\rangle : \tau' \to \tau''}$$

$$\frac{}{\Gamma, x : \tau \vdash x : \tau}$$

$$\frac{\Gamma, y : \tau_2 \vdash t : \tau \quad \Gamma \vdash s : \tau_1}{\Gamma, x : \tau_1 \to \tau_2 \vdash t[xs/y] : \tau} \qquad \frac{\Gamma, x : \tau' \vdash t : \tau''}{\Gamma, \Delta \vdash \langle x \cdot t \rangle : \tau' \to \tau''} \ (\Gamma \text{ arrowed}) \qquad \frac{\Gamma \vdash t : \tau \quad \Gamma, x : \tau \vdash s : \tau'}{\Gamma \vdash s[t/x] : \tau'} \ (Cut)$$

Figure 6: Sequent calculus for near-intuitionistic logic

where Γ is restricted to be an *arrowed* context, i.e. one mapping each variable x_i to an arrow type $\tau_i' \to \tau_i''$, and Δ is any context (used to build in weakening); and of a rule (Cut) to type (implicit) substitutions:

$$\frac{\Gamma \vdash t : \tau \quad \Gamma, x : \tau \vdash s : \tau'}{\Gamma \vdash s[t/x] : \tau'} \ (Cut)$$

The latter is the famous cut rule of sequent calculi, and is easily shown to be admissible in the system of Figure 5, by structural induction on s (see Appendix A).

Our first point is to show that the natural deduction system, i.e. the typing system of Figure 5 (considering types as formulae, and λ_{clos}-terms as proof-terms), is the proof-theoretical counterpart of the semantics of Definition 1:

Theorem 1 *The deduction rules of Figure 5 are sound and complete for the semantics of Definition 1.*

Proof: I.e., given a context $x_1 : \tau_1, \ldots, x_n : \tau_n$ and a type τ_0, there is a λ_{clos}-term t such that $x_1 : \tau_1, \ldots, x_n : \tau_n \vdash t : \tau_0$ is derivable in the system of Figure 5 if and only if for every near-intuitionistic frame $(\mathcal{W}, \leq, \rho)$, for every $w \in W$ such that $w \models \tau_1$ and $\ldots$ and $w \models \tau_n$, $w \models \tau_0$ (where $\models$ denotes $\models_{(\mathcal{W},\leq,\rho)}$). The proof uses standard arguments: see Appendix B for a full proof.

Soundness: by structural induction on t. The only non-trivial case is for closures $\langle x \cdot s, \{x_1 := t_1, \ldots, x_k := t_k\}\rangle$: soundness works in this case because any true *arrow* type must remain true in all later worlds.

Completeness: by an argument à la Smullyan [GLM97]. Assume that there is no λ_{clos}-term t such that $\Gamma_0 \vdash t : \tau_0$ is derivable. We build a frame and a world w_0 such that w_0 satisfies Γ_0 but $w_0 \not\models \tau_0$ as follows. First, consider *signed formulas* F, which are either positive $+\tau$ or negative $-\tau$, where τ is a type. (Intuitively, $+\tau$ means that τ is assumed true, and $-\tau$ means that τ is assumed false; we do this because we have no classical negation symbol in our language of formulae.) A set S of signed formulas is *consistent* if and only if for every finite set of positive formulas $+\tau_1, \ldots, +\tau_n$ (not necessarily distinct) and every negative formula $-\tau$ in S, there is no λ_{clos}-term t such that $x_1 : \tau_1, \ldots, x_n : \tau_n \vdash t : \tau$. By Zorn's Lemma, every consistent set is contained in some maximally consistent set. We build the desired frame $(\mathcal{W}, \leq, \rho)$ by letting the worlds w be all maximally consistent sets of signed formulae, $w \leq w'$ if and only if, for every positive formula of the form $+(\tau_1 \to \tau_2)$ in w such that $+\tau_1$ is in w', then $+\tau_2$ is in w', and we let ρ map every base type b to the set of worlds w such that $+b \in w$. We also define the counter-example world w_0 as any maximally consistent set containing $\{+\tau_1, \ldots, +\tau_n, -\tau_0\}$ (which is consistent by assumption). Standard arguments show that $w \models \tau$ if and only if $+\tau \in w$, if and only if $-\tau \notin w$: so $w_0 \not\models \tau_0$. $\Box$

The point now is that the reduction rules of λ_{clos} encode cut-elimination in near-intuitionistic logic. To make this precise, consider the Gentzen system of Figure 6, where we decorate types on the left with variables and types on the right with λ_{clos}-terms, to show the correspondence with the natural deduction system of Figure 5.

Theorem 2 (Cut Elimination) $\Gamma \vdash s : \tau$ *is derivable in the system of Figure 5 if and only if it is derivable in the system of Figure 6, if and only some sequent* $\Gamma \vdash t : \tau$ *is derivable in the system of Figure 6 without* (Cut)*, with* $s \longrightarrow^* t$ *in* λ_{clos}.

$$
\begin{array}{rcl}
x^* & =_{\mathrm{df}} & x \\
(st)^* & =_{\mathrm{df}} & S_0(s^*, t^*) \\
(\langle x \cdot s, \{x_1 := s_1, \ldots, x_k := s_k\}\rangle)^* & =_{\mathrm{df}} & \\
 & & ([x]s^*)[s_1^*/x_1, \ldots, s_k^*/x_k]
\end{array}
\qquad
\begin{array}{rcll}
[x]x & =_{\mathrm{df}} & I_0 & \\
[x]y & =_{\mathrm{df}} & K_0(y) & (y \neq x) \\
[x]I_\ell & =_{\mathrm{df}} & I_{\ell+1} & \\
[x]S_\ell(u,v) & =_{\mathrm{df}} & S_{\ell+1}([x]u, [x]v) & \\
[x]K_\ell(u) & =_{\mathrm{df}} & K_{\ell+1}([x]u) &
\end{array}
$$

Figure 7: Translation from λ_{clos} to SKInT

Proof: The first equivalence is easy and fairly standard. The second equivalence follows from the following facts: subject reduction holds for simply typed λ_{clos}-reduction; simply typed λ_{clos}-reduction is strongly normalizing; and every simply typed λ_{clos}-normal term is the proof term of some cut-free proof in the system of Figure 6.

To show that simply-typed λ_{clos}-terms are strongly normalizing, a simple way is to translate λ_{clos}-terms to λ-terms by letting $h(x) =_{\mathrm{df}} x$, $h(st) =_{\mathrm{df}} h(s)h(t)$, $h(\langle x \cdot s, \{x_1 := t_1, \ldots, x_k := t_k\}\rangle) =_{\mathrm{df}} \lambda x \cdot s[h(t_1)/x_1, \ldots, h(t_k)/x_k]$: if $s \longrightarrow^* t$ in λ_{clos}, then $s \longrightarrow^* t$ in the λ-calculus. For every strongly normalizing λ-term u, let $\nu(u)$ denote the length of its longest reduction sequence. Then we show that for every integer n, $P(n)$ implies $P(n+1)$, where $P(n)$ is the property that for every λ_{clos}-term s such that $\nu(h(s')) < n$ for every subterm s' of s, s is strongly normalizing. This, in turn, is proved by structural induction on s: if s is an application $s_1 s_2$ and $\nu(h(s)) < n+1$, then we show that s is strongly normalizing by induction on the sum of the longest λ_{clos}-reductions starting from s_1 and s_2 respectively; if $s = \langle x \cdot t, \{x_1 := t_1, \ldots, x_k := t_k\}\rangle$ and $\nu(h(s)) < n+1$, then this is by double induction on the sum of the lengths of the longest λ_{clos}-reductions starting from t, t_1, …, t_k first, then on the sum of the sizes of t_1, …, t_k; the other cases are clear. It follows (by induction on n this time) that $P(n)$ holds for all n; since every subterm s' of a simply-typed λ_{clos}-term s is again simply-typed, and $h(s')$ is a simply-typed (therefore strongly normalizing) λ-term, $P(n)$ holds for $n =_{\mathrm{df}} \max\{\nu(h(s')) | s' \text{ subterm of } s\}$, so s is strongly normalizing in λ_{clos}.

To show that every simply-typed λ_{clos}-normal term is the proof term of some cut-free proof, notice that every application subterm is of the form $xs_1 \ldots s_n$, where x is a variable, and that (by rule (ι)) in a closure $\langle x \cdot t, \theta\rangle$, θ may only map variables to application terms (not closures). The main difficulty is that $\langle x \cdot t, \theta\rangle = \langle x \cdot t\rangle\hat{\theta}$, and that applying the substitution $\hat{\theta}$ must be done without using the (Cut) rule. Since θ only maps variables to applications, we can do this by using the left introduction rule for arrow types instead (see Figure 6), e.g. $\langle x \cdot t, \{x_1 := z_1 t_1\}\rangle$ gives rise to the following piece of proof (where Γ and τ_1'' are arrowed):

$$
\dfrac{\dfrac{\vdots}{\Gamma \vdash t_1 : \tau_1'} \qquad \dfrac{\dfrac{\vdots}{\Gamma, x_1 : \tau_1'', x : \tau' \vdash t : \tau''}}{\Gamma, x_1 : \tau_1'' \vdash \langle x \cdot t\rangle : \tau' \to \tau''}}{\Gamma, z_1 : \tau_1' \to \tau_1'' \vdash \langle x \cdot t\rangle[z_1 t_1/x_1] : \tau' \to \tau''}
$$

See Appendix C for a more complete proof, of these results and the facts that λ_{clos} and $\lambda_{\mathrm{clos}\,\eta}$ are both confluent, enjoy finite developments, and that the simply typed $\lambda_{\mathrm{clos}\,\eta}$-calculus is strongly normalizing as well. □

Now we come (at last) to the reason why we introduced λ_{clos} in the first place. The following theorem shows in particular that SKInT is a complete proof-term language for near-intuitionistic logic, which was the main point of this section.

Theorem 3 *Let $s \mapsto s^*$ be the map from λ_{clos}-terms to* SKInT*-terms shown in Figure 7. Conversely, let $u \mapsto u^\circ$ be the map from* SKInT*-terms to λ_{clos}-terms of Figure 8. Then:*

(i) If $\Gamma \vdash s : \tau$ is derivable in the system of Figure 5, then $\Gamma \vdash s^* : \tau$ is derivable in the system of Figure 4;

(ii) If $\Gamma \vdash u : \tau$ is derivable in the system of Figure 4, then $\Gamma \vdash u^\circ : \tau$ is derivable in the system of Figure 5;

$$
\begin{array}{rcl}
x^\circ & =_{\rm df} & x \\
I_\ell{}^\circ & =_{\rm df} & \langle x_0 \cdot \langle x_1 \cdot \ldots \langle x_{\ell-1} \cdot \\
 & & \quad \langle z \cdot z \rangle \rangle \ldots \rangle \rangle \\
(S_\ell(u,v))^\circ & =_{\rm df} & \mathcal{S}_\ell(u^\circ, v^\circ) \\
(K_\ell(u))^\circ & =_{\rm df} & \mathcal{K}_\ell(u^\circ)
\end{array}
\qquad
\begin{array}{rcl}
\mathcal{S}_0(s,t) & =_{\rm df} & st \\
\mathcal{S}_{\ell+1}(s,t) & =_{\rm df} & \langle z \cdot \mathcal{S}_\ell(xz, yz), \{x := s, y := t\}\rangle \\
 & & \\
\mathcal{K}_0(s) & =_{\rm df} & \langle z \cdot x, \{x := s\}\rangle \\
\mathcal{K}_{\ell+1}(s) & =_{\rm df} & \langle z \cdot \mathcal{K}_\ell(xz), \{x := s\}\rangle
\end{array}
$$

Figure 8: A translation from SKInT to $\lambda_{\rm clos}$

(*iii*) *If* $s \longrightarrow t$ *in* $\lambda_{\rm clos}$ *(resp.* $\lambda_{{\rm clos}\eta}$*), then* $s^* \longrightarrow^* t^*$ *in* SKInT *(resp.* SKInT_η*);*

(*iv*) *If* $u \longrightarrow^* v$ *in* SKInT*, resp.* SKInT_η*, then* $u^\circ \approx v^\circ$ *in* $\lambda_{{\rm clos}\eta}$*, where* $\approx$ *denotes convertibility;*

(*v*) *For every* $\lambda_{\rm clos}$*-term* s*,* $s^{*\circ} \approx s$ *in* $\lambda_{\rm clos}$*;*

(*vi*) *For every* SKInT*-term* u*,* $u^{\circ *} \longrightarrow_\eta^* u$ *in* SKInT_η*;*

(*vii*) *For every* $\lambda_{\rm clos}$*-normal (resp.* $\lambda_{{\rm clos}\eta}$*-normal)* $\lambda_{\rm clos}$*-term* s*,* s^* *is a* SKInT*-normal (resp.* SKInT_η*-normal)* SKInT*-term.*

Proof: Tedious series of computations (see Appendix D). □

In other words, (*iii*)–(*vi*) tell us that, up to $\lambda_{{\rm clos}\eta}$-convertibility on one side, and up to SKInT_η-convertibility on the other, the $\lambda_{\rm clos}$-terms and the SKInT-terms are isomorphic (the isomorphism being given by the pair of inverse functions $s \mapsto s^*$ and $u \mapsto u^\circ$). This isomorphism also preserves types in both directions, by (*i*) and (*ii*). And one half of it (the map $s \mapsto s^*$) even preserves normal forms literally, by (*vii*).

In particular, the typing rules of SKInT are sound and complete for near-intuitionistic logic, and SKInT-normal forms correspond to cut-free proofs.

3 Conjunctive Types

We now turn to the relationship between conjunctive type systems for SKInT and termination properties, à la Sallé-Coppo [CC90]. We first recall a few notions from [GGL98]. Define the *spines* S by the grammar:

$$S ::= x \mid I_\ell \mid S_\ell S \mid K_\ell S \qquad (\ell \geq 0)$$

The *arity* of a spine is the number of operators of the form S_ℓ, $\ell \geq 0$, in it. If n is the arity of S, and v_1, ..., v_n are n SKInT-terms, then the term $S[v_1, \ldots, v_n]$ is defined by:

$$
\begin{array}{ll}
x[] =_{\rm df} x & I_\ell[] =_{\rm df} I_\ell \\
(S_\ell S)[v_1, \ldots, v_n] & =_{\rm df} S_\ell(S[v_1, \ldots, v_{n-1}], v_n) \\
(K_\ell S)[v_1, \ldots, v_n] & =_{\rm df} K_\ell(S[v_1, \ldots, v_n])
\end{array}
$$

Every term u can be written in a unique way as $S[v_1, \ldots, v_n]$: the spine S is the sequence of operators occurring along the leftmost branch of u, read top-down. Then the terms v_1, ..., v_n are the arguments of operators of the form S_ℓ, $\ell \geq 0$, on the spine, read bottom-up. The terms v_1, ..., v_n are called the *arguments* of u. Iterating this decomposition of terms in spine and arguments allows us to see terms as trees of spines.

Call a *one-step spine reduction* $u \longrightarrow^s v$ any one-step reduction of a redex occurring on the spine of u. A *spine reduction* $u \longrightarrow^{s*} v$ is a sequence of one-step spine reductions. A term that has no one-step spine contractum is called *spine-normal*. Spine reductions play the role of head reductions in the λ-calculus. (However, spine reductions are not unique.)

Define *standard reductions* $u \longrightarrow^{std*} v$ by induction on v viewed as a tree of spines, if and only if $u \longrightarrow^{s*} S[u_1, \ldots, u_n]$, and $v = S[v_1, \ldots, v_n]$, where $u_i \longrightarrow^{std*} v_i$ for each i, $1 \leq i \leq n$. SKInT standardizes [GGL98], in that $u \longrightarrow^* v$ (in SKInT) implies $u \longrightarrow^{std*} v$. In particular, a term is weakly normalizable if and only if it has a normalizing standard reduction.

Definition 2 *The* SKInT*-term u is* solvable *if any of the following equivalent conditions hold:*

(i) *All* SKInT*-spine reductions starting from u terminate;*

(ii) *Some* SKInT*-spine reduction starting from u terminates.*

Proof: That (i) implies (ii) is clear. Conversely, write $u \rightsquigarrow v$ when u is ΣT-normal, has a spine βI-redex $S_\ell(I_\ell, w)$, and v is the unique ΣT-normal form of the spine βI-contraction of u. (Notice that any term has at most one spine βI-redex, hence $\rightsquigarrow$-reductions are unique.) By examining how rules commute, we can show that if $u \longrightarrow^{s*} v$ by using βI n times, then $\Sigma T(u) \rightsquigarrow^* \Sigma T(v)$ in exactly n steps (see Appendix E). It follows that, if (ii) holds (with termination in n spine βI-steps), then any $\rightsquigarrow$-reduction starting from u terminates (in exactly n steps), and therefore all spine reductions do exactly n spine βI-steps; as ΣT terminates, all these spine reductions must be finite. $\Box$

We wish to characterize solvable, weakly normalizing and strongly normalizing as terms that can be typed in some systems of conjunctive types. It will be profitable to use a simplified format for conjunctive types, inspired from Émilie Sayag's Ph.D. thesis [Say97]. Define the *simple intersection types* τ by:

$$\tau ::= B \mid \mu \to \tau \qquad \mu ::= [\tau_1, \ldots, \tau_n] \qquad (n \geq 0)$$

where $[\tau_1, \ldots, \tau_n]$ is the multiset of types $\tau_1, \ldots, \tau_n$. Intuitively, $[\tau_1, \ldots, \tau_n]$ denotes the intersection of $\tau_1, \ldots, \tau_n$. If $n = 0$, $[]$ denotes the set of all terms. We shall also write ω for $[]$, and $\mu_1 \wedge \mu_2$ for the multiset union of μ_1 and μ_2.

The typing rules of Sayag's system are, to take an analogy with linear logic, in multiplicative style (i.e., the Γ parts of the premises are merged to get that of the conclusion in each rule). We shall use the slightly different additive-style system of Figure 9, which we name $\mathcal{S}\omega$. It is easy to see that both are equivalent in the following sense. First, if $\vdash_{s[]} u : A \Rightarrow \tau$ in Sayag's system (where A is a total function from variables to μ-types, mapping all but finitely many to $[]$), then $\{x : A(x) \mid A(x) \neq []\} \vdash u : \tau$ in $\mathcal{S}\omega$. Conversely, if $\Gamma \vdash u : \tau$ in system $\mathcal{S}\omega$, then $\Gamma \vdash u : \tau$ is the weakening of some sequent $\Gamma' \vdash u : \tau'$ such that $\vdash_{s[]} \{x \mapsto \mu \mid x : \mu \in \Gamma'\} \cup \{x \mapsto [] \mid x \notin \Gamma'\} \Rightarrow \tau'$ in Sayag's system. (Call μ a strengthening of μ' if and only if $\mu = \mu' \wedge \mu''$ for some μ''; call τ a weakening of τ' if and only if either $\tau = \mu_0 \to \ldots \to \mu_{n-1} \to b$, $\tau' = \mu'_0 \to \ldots \to \mu'_{n-1} \to b$, and μ_i is a strengthening of μ'_i for each i, $0 \leq i < n$; call the sequent $\Gamma \vdash u : \tau$ a weakening of $\Gamma' \vdash u : \tau'$ if and only if for every binding $x : \mu'$ in Γ', there is a binding $x : \mu$ in Γ with μ strengthening μ', and τ is a weakening of τ'.)

We do this change of system to have a simpler set of typing rules. Doing this, we lose one important property of Sayag's system: Sayag's typings are always minimal, in the sense that derivable sequents contain only relevant information about the terms being typed. In this paper, we don't care about that aspect of typing. Moreover, this change of system will allow us to keep the same system (that of Figure 9) even to characterize strongly normalizing terms (we shall just change the set of types), contrarily to Sayag.

Strongly normalizing λ-terms are those that are typable in system $\mathcal{S}$, defined as follows. Call $\mathcal{S}$*-types* the types generated by the grammar:

$$\tau ::= B \mid \mu \to \tau \qquad \mu ::= [\tau_1, \ldots, \tau_n] \qquad (n \geq 1)$$

In other words, multisets of types are now restricted to be non-empty. We define system $\mathcal{S}$ as the system of Figure 9 (again), but where τ and μ-types are restricted to be $\mathcal{S}$-types.

Correspondingly, we endow SKInT with the typing rules of Figure 10, yielding a typing system that we call $S\omega$ (when types are $\mathcal{S}\omega$-types), or S (when types are $\mathcal{S}$-types).

Call an $\mathcal{S}\omega$-type τ *definite positive* if and only if ω only occurs negatively in τ. More formally, the definite positive types τ^+ and the definite negative types τ^- are defined by the grammar:

$$\begin{array}{llll} \tau^+ & ::= B \mid \mu^- \to \tau^+ & \mu^+ & ::= [\tau_1^+, \ldots, \tau_n^+] \quad (n \geq 1) \\ \tau^- & ::= B \mid \mu^+ \to \tau^- & \mu^- & ::= [\tau_1^-, \ldots, \tau_n^-] \quad (n \geq 0) \end{array}$$

A context Γ is *definite negative* if and only if every binding in Γ is of the form $x : \mu^-$ with μ^- definite negative. We say that the typing judgement $\Gamma \vdash s : \tau$ is *definite positive* if and only if Γ is definite negative and τ is definite positive.

We first have the following, which we leave to the reader to check:

$$\frac{}{\Gamma, x : \mu \wedge \tau \vdash x : \tau}$$

$$\frac{\Gamma \vdash u : [\tau_1, \ldots, \tau_n] \to \tau \quad \Gamma \vdash v : \tau_1 \quad \ldots \quad \Gamma \vdash v : \tau_n}{\Gamma \vdash uv : \tau} \qquad \frac{\Gamma, x : \mu \vdash u : \tau}{\Gamma \vdash \lambda x \cdot u : \mu \to \tau}$$

Figure 9: Conjunctive types for the λ-calculus

$$\frac{}{\Gamma, x : \mu \wedge \tau \vdash x : \tau} \qquad \frac{}{\Gamma \vdash I_\ell : \mu_0 \to \ldots \to \mu_{\ell-1} \to \mu_\ell \wedge \tau \to \tau}$$

$$\frac{\begin{array}{l}\Gamma \vdash u : \mu_0 \to \ldots \to \mu_{\ell-1} \to [\tau_1, \ldots, \tau_n] \to \tau \\ \Gamma \vdash v : \mu_0 \to \ldots \to \mu_{\ell-1} \to \tau_1 \\ \ldots \\ \Gamma \vdash v : \mu_0 \to \ldots \to \mu_{\ell-1} \to \tau_n\end{array}}{\Gamma \vdash S_\ell(u, v) : \mu_0 \to \ldots \to \mu_{\ell-1} \to \tau} \qquad \frac{\begin{array}{r}\Gamma \vdash u : \mu_0 \to \ldots \to \mu_{\ell-1} \\ \to \mu_\ell \to \tau\end{array}}{\begin{array}{r}\Gamma \vdash K_\ell(u) : \mu_0 \to \ldots \to \mu_{\ell-1} \\ \to \mu \to \mu_\ell \to \tau\end{array}}$$

Figure 10: The system of conjunctive types for SKInT

Lemma 4 (Subject Reduction) *If $\Gamma \vdash u : \tau$ in $S\omega$, resp. S, and $u \longrightarrow^* v$ in* SKInT *or* SKInT_η, *then $\Gamma \vdash v : \tau$ in $S\omega$, resp. S.*

Theorem 5 (Normalization) *The following holds:*

(i) If $\Gamma \vdash u : \tau$ is derivable in system $S\omega$, then u is solvable;

(ii) If $\Gamma^- \vdash u : \tau^+$ is definite positive and derivable in system $S\omega$, then u is weakly normalizing in SKInT, *resp. in* SKInT_η;

(iii) If $\Gamma \vdash u : \tau$ is derivable in system S, then u is strongly normalizing in SKInT, *resp. in* SKInT_η.

Proof: The idea is to translate the term u to a λ-term, and to use the corresponding results for the λ-calculus. The natural choice for the translation is $u \mapsto [\![u]\!](s_0, \ldots, s_{n-1})$ for some choice of sequence $s_0, \ldots, s_{n-1}$ of λ-terms. However, some reductions in SKInT would then translate to equalities in the λ-calculus, which would prevent us from drawing the desired conclusions. (Consider e.g. $u =_{\mathrm{df}} S_0(K_0(u_1), u_2)$: whatever the reductions in u_2, they won't be taken into account by the translation, since $[\![u]\!](s_0, \ldots, s_{n-1}) = [\![u_1]\!](s_0, \ldots, s_{n-1})$ does not depend on u_2 in any way.)

For this reason, we modify this translation to map SKInT-terms to $\lambda_{\oplus\epsilon}$-*terms*: the idea is that instead of dropping some arguments (like s_ℓ in the definition of $[\![K_\ell(u)]\!](s_0, \ldots, s_{n-1})$ for $n \geq \ell$), we shall keep them on the left of some binary operator $\oplus$ (instead of $s_{\ell+1}$, we shall write $s_\ell \oplus s_{\ell+1}$, see Figure 11 below) such that $s \oplus t$ is semantically equivalent to t alone.

The $\lambda_{\oplus\epsilon}$-calculus is defined by the grammar:

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

and its reduction rules are $\beta\eta$-reduction plus:

$$(\epsilon) \quad \epsilon t \to t \qquad (\oplus-) \quad t_1 \oplus t_2 \to t_2 \qquad (\oplus) \quad (t_1 \oplus t_2) \oplus t_3 \to t_1 \oplus (t_2 \oplus t_3)$$

We define type systems that we call again $\mathcal{S}\omega$, resp. $\mathcal{S}$, defined on $\mathcal{S}\omega$-types, resp. $\mathcal{S}$-types, and whose typing rules are those of Figure 9, plus:

$$\frac{\Gamma \vdash t : \tau}{\Gamma \vdash \epsilon t : \tau} \qquad \frac{[\Gamma \vdash t_1 : \tau_1] \quad \Gamma \vdash t_2 : \tau_2}{\Gamma \vdash t_1 \oplus t_2 : \tau_2}$$

$$[\![u]\!]_\oplus(s_0,\ldots,s_{n-1}) =_{\text{df}} \begin{cases} xs_0\ldots s_{n-1} & \text{if } u = x \\ \epsilon(s_0 \oplus \ldots \oplus s_{\ell-1} \oplus s_\ell)s_{\ell+1}\ldots s_{n-1} & \text{if } u = I_\ell \\ [\![v]\!]_\oplus(s_0,\ldots,s_{\ell-1}, s_\ell \oplus s_{\ell+1}, s_{\ell+2},\ldots,s_{n-1}) & \text{if } u = K_\ell(v) \\ [\![v]\!]_\oplus(s_0,\ldots,s_{\ell-1}, \\ \qquad [\![w]\!]_\oplus(s_0,\ldots,s_{\ell-1}), s_\ell,\ldots,s_{n-1}) & \text{if } u = S_\ell(v,w) \end{cases}$$
$$\text{when } n > \dim u$$

$$[\![u]\!]_\oplus(s_0,\ldots,s_{n-1}) =_{\text{df}} \lambda x_n \cdot \ldots \cdot \lambda x_{m-1} \cdot [\![u]\!]_\oplus(s_0,\ldots,s_{n-1},x_n,\ldots,x_{m-1})$$
$$\text{when } n < m = \dim u + 1$$

Figure 11: Interpretation of SKInT-terms as $\lambda_{\oplus\epsilon}$-terms

where the bracketed premise $\Gamma \vdash t_1 : \tau_1$ is included in $\mathcal{S}$, but omitted in $\mathcal{S}\omega$.

There is an erasing translation $t \mapsto |t|$ from $\lambda_{\oplus\epsilon}$ to the λ-calculus (with β-reduction): $|\epsilon t| = |t|$, $|t_1 \oplus t_2| = |t_2|$, $|x| = x$, $|t_1 t_2| = |t_1|\ |t_2|$, $|\lambda x \cdot tx| = |t|$ if x is not free in t, and $|\lambda x \cdot t| = \lambda x \cdot |t|$ if t is not of the form $t'x$ with x not free in t'. $\lambda_{\oplus\epsilon}$ has the subject reduction property, and for every $\lambda_{\oplus\epsilon}$-term t:

(a) If $\Gamma \vdash t : \tau$ in $\mathcal{S}\omega$, then t is solvable, i.e., all head-reductions starting from t terminate. We call *head-reduction* in $\lambda_{\oplus\epsilon}$ any (ϵ), $(\oplus-)$, $(\oplus)$ or (η)-step, or any (β)-reduction step $s \longrightarrow t$ such that $|s| \longrightarrow |t|$ by a head (β)-reduction step in the λ-calculus (in particular, the erasing translation does not erase the $\lambda_{\oplus\epsilon}$-redex).

(b) If $\Gamma \vdash t : \tau$ in $\mathcal{S}$, then t is strongly normalizing.

The proofs are by appealing to the same properties in the λ-calculus, using the erasing translation above (see Appendix F), or by reducibility methods (for (b)): see [GL97].

Define the *dimension* $\dim u$ of a SKInT-term u by:

$$\begin{array}{rcl@{\quad}rcl} \dim x & =_{\text{df}} & -1 & \dim I_\ell & =_{\text{df}} & \ell \\ \dim K_\ell(u) & =_{\text{df}} & \max(\ell, \dim u) + 1 & \dim S_\ell(u,v) & =_{\text{df}} & \max(\ell, \dim u) - 1 \end{array}$$

The new translation $u \mapsto [\![u]\!]_\oplus(s_0,\ldots,s_{n-1})$, parameterized by a list $s_0, \ldots, s_{n-1}$ of $\lambda_{\oplus\epsilon}$-terms, is described in Figure 11.

By abuse of language, say that $\Gamma \vdash s : \mu$ is derivable in $\mathcal{S}\omega$, resp. $\mathcal{S}$, where $\mu = [\tau_1,\ldots,\tau_k]$, if and only if $\Gamma \vdash s : \tau_i$ is derivable in $\mathcal{S}\omega$, resp. $\mathcal{S}$, for every i, $1 \leq i \leq k$. An easy structural induction on u shows that, if $\Gamma \vdash u : \mu_0 \to \ldots \to \mu_{n-1} \to \tau$ in $S\omega$, resp. S, and $\Gamma \vdash s_i : \mu_i$ in $\mathcal{S}\omega$, resp. $\mathcal{S}$, for every i, $0 \leq i < n$, then $\Gamma \vdash [\![u]\!]_\oplus(s_0,\ldots,s_{n-1}) : \tau$ in $\mathcal{S}\omega$, resp. $\mathcal{S}$.

A tedious check now shows that whenever $u \longrightarrow v$ in SKInT$_\eta$, then for every sequence $s_0, \ldots, s_{n-1}$ of $\lambda_{\oplus\epsilon}$-terms, $[\![u]\!]_\oplus(s_0,\ldots,s_{n-1}) \longrightarrow^* [\![v]\!]_\oplus(s_0,\ldots,s_{n-1})$ in $\lambda_{\oplus\epsilon}$, resp. $\longrightarrow^+$ in the case of (SI_ℓ), (ηS_ℓ) and a few other rules (see Appendix G).

By (b), any reduction R in SKInT, resp. SKInT$_\eta$, starting from a typable term in system S uses only finitely many instances of rules (SI_ℓ) and (ηS_ℓ), $\ell \geq 0$. But since ΣT terminates, there are finitely many reduction steps (in ΣT) inbetween two (SI_ℓ) or (ηS_ℓ)-steps. So R is finite, proving (iii).

To show (i), first notice that if $u \longrightarrow^s v$ in SKInT (or SKInT$_\eta$), then $[\![u]\!]_\oplus(s_0,\ldots,s_{n-1}) \longrightarrow^* [\![v]\!]_\oplus(s_0,\ldots,s_{n-1})$ (resp. $\longrightarrow^+$ if $u \longrightarrow^s v$ by (SI_ℓ), (ηS_ℓ) and a few other rules) by head-reductions in $\lambda_{\oplus\epsilon}$(see Appendix G, Claim (h)) . By (a), any spine-reduction step in SKInT, resp. SKInT$_\eta$, starting from a typable term in $S\omega$ has only spine-reductions that use finitely many instances of (SI_ℓ) or (ηS_ℓ), $\ell \geq 0$. Since (ΣT) terminates, (i) follows.

To show (ii), we would like to use a similar argument, but any $\lambda_{\oplus\epsilon}$-term with a definite positive typing normalizes only weakly, and then we need to show that we can lift back the given normalization strategy in $\lambda_{\oplus\epsilon}$ to some normalization strategy in SKInT, and this is not easy. Instead, we observe the following. Let u' be a spine-normal SKInT-term, and write u' as $S[u_1,\ldots,u_k]$. We may then write S as a word of the form

$K_{i_{01}} \dots K_{i_{0n_0}} S_{j_1} K_{i_{11}} \dots K_{i_{1n_1}} S_{j_2} \dots S_{j_k} K_{i_{k1}} \dots K_{i_{kn_k}} L$, with $L = I_j$ or L a variable (in which case we let $j =_{\mathrm{df}} -1$), with:

$$i_{01} > \dots > i_{0n_0} \geq j_1 > i_{11} > \dots > i_{1n_1} \geq j_2 > \dots \geq j_k > i_{k1} > \dots > i_{kn_k} > j \geq -1$$

and $k \geq 0$, $n_0 \geq 0$, $n_1 \geq 0$, $\dots$, $n_k \geq 0$, and when $n_i = 0$, the notation $j_i > i_{i1} > \dots > i_{in_i} \geq j_{i+1}$ means $j_i \geq j_{i+1}$. If $n > \dim u'$, then we have:

$$\begin{aligned}
&|[\![u']\!]_\oplus(s_0, \dots, s_{n-1})| \\
&= L'|s_{j+1}| \dots |\widehat{s_{i_{kn_k}}}| \dots |\widehat{s_{i_{k1}}}| \dots |s_{j_k-1}| \\
&\quad |[\![u_1]\!]_\oplus(s_0, \dots, s_{j_k-1})||s_{j_k}| \dots |\widehat{s_{i_{(k-1)n_{k-1}}}}| \dots |\widehat{s_{i_{(k-1)1}}}| \dots |s_{j_{k-1}-1}| \\
&\quad |[\![u_2]\!]_\oplus(s_0, \dots, s_{j_{k-1}-1})| \dots \\
&\quad \dots \\
&\quad |[\![u_k]\!]_\oplus(s_0, \dots, s_{j_1-1})||s_{j_1}| \dots |\widehat{s_{0n_0}}| \dots |\widehat{s_{01}}| \dots s_{n-1}
\end{aligned} \tag{1}$$

where $L' = |s_j|$ if $L = I_j$, or $L' = x$ if L is a variable x, and $s_i \dots \hat{s}_{i_1} \dots \hat{s}_{i_p} \dots s_j$ denotes the sequence of terms s_i, $\dots$, s_j from which the terms s_{i_1}, $\dots$, s_{i_p} have been omitted. If $n \leq \dim u'$, then $|[\![u']\!]_\oplus(s_0, \dots, s_{n-1})|$ is the η-normal form of $\lambda s_n \cdot \dots \cdot \lambda s_{\dim u'} \cdot |[\![u']\!]_\oplus(s_0, \dots, s_{\dim u'})|$, and is therefore of the form $\lambda s_n \cdot \dots \cdot \lambda s_{m-1} \cdot |[\![u']\!]_\oplus(s_0, \dots, s_{m-1})|$.

As far as types are concerned, let u' have a definite positive typing. Any (definite positive) $S\omega$-type of u' must be of the form $\mu_0 \to \dots \to \mu_{i_{01}} \to \mu_{i_{01}+1} \to \tau$, and the typing derivation leading to this type must map each u_i, $1 \leq i \leq k$, to the types $\mu_0 \to \dots \to \mu_{j_{n+1-i}-1} \to \tau_{ip}$, $1 \leq p \leq m_i$, for some types τ_{ip}. Let μ'_i be $[\tau_{i1}, \dots, \tau_{im_i}]$. Then this typing derivation also gave L the type:

$$\begin{aligned}
&\mu_0 \to \dots \to \hat{\mu}_{i_{kn_k}} \to \dots \to \hat{\mu}_{i_{k1}} \to \dots \to \mu_{j_k-1} \\
&\to \mu'_1 \to \mu_{j_k} \to \dots \to \hat{\mu}_{i_{(k-1)n_{k-1}}} \to \dots \to \hat{\mu}_{i_{(k-1)1}} \to \dots \to \mu_{j_{k-1}-1} \\
&\to \dots \\
&\to \mu'_{k-1} \to \mu_{j_2} \to \dots \to \hat{\mu}_{i_{1n_1}} \to \dots \to \hat{\mu}_{i_{11}} \to \dots \to \mu_{j_1-1} \\
&\to \mu'_k \to \mu_{j_1} \to \dots \to \hat{\mu}_{i_{0n_0}} \to \dots \to \hat{\mu}_{i_{01}} \to \mu_{i_{01}+1} \to \tau
\end{aligned} \tag{2}$$

If L is a variable, since the typing context is definite negative, the type above must be definite negative, so in particular every μ'_i, $1 \leq i \leq k$, is definite positive. Since the type of u' was assumed definite positive, every μ_j is definite negative, so the types $\mu_0 \to \dots \to \mu_{j_{n+1-i}-1} \to \tau_{ip}$ of u_i are definite positive. Similarly, if L is I_j, then recall that $j < i_{kn_k}$ (if $n_k \neq 0$) or $j < j_k$ (if $n_k = 0$ and $k \neq 0$), and by the form of the typing rule for I_j, every μ'_i, $1 \leq i \leq k$ must occur negatively in μ_ℓ, hence is definite positive (if $k \neq 0$; this is trivial if $k = 0$), hence again the types assigned to u_i are all definite positive.

Having made these remarks, let u be a SKInT-term with a definite positive $S\omega$-typing. Then $|[\![u]\!]_\oplus(s_0, \dots, s_{n-1})|$ is a λ-term with a definite positive $S\omega$-typing, for any sequence s_0, $\dots$, s_{n-1} of the right types, hence it β-normalizes weakly. Recall that a weakly normalizing λ-term t has a finite Böhm tree: let $h(t)$ be the height of this tree. We show that, under the assumption that u has a definite positive $S\omega$-typing and that $|[\![u]\!]_\oplus(x_0, \dots, x_{n-1})|$ normalizes weakly for some sequence of variables x_0, $\dots$, x_{n-1}, then u SKInT-normalizes weakly. This is by induction on $h(t)$, where $t = |[\![u]\!]_\oplus(x_0, \dots, x_{n-1})|$. First, by (i) and since u has an $S\omega$-typing, u is SKInT-solvable: let $u' =_{\mathrm{df}} S[u_1, \dots, u_k]$ be any spine-normal form of u. Since $u \longrightarrow^{s*} u'$, as in case (i), $|[\![u]\!]_\oplus(x_0, \dots, x_{n-1})|$ head-rewrites to $|[\![u']\!]_\oplus(x_0, \dots, x_{n-1})|$, and the latter is head-normal, since x_j is a variable (by inspection of Equation 1). By the remark on the types of spine-normal forms (Equation 2), each u_i, $1 \leq i \leq k$, also has a definite positive typing. Let now t_i be $|[\![u_i]\!]_\oplus(x_0, \dots, x_{j'_{k+1-i}})|$: by Equation 1, $h(t_i) < h(|[\![u']\!]_\oplus(x_0, \dots, x_{n-1})|) = h(|[\![u]\!]_\oplus(x_0, \dots, x_{n-1})|)$. So the induction hypothesis applies: each u_i SKInT-normalizes weakly, say to some term v_i. Therefore u SKInT-normalizes weakly to $S[v_1, \dots, v_k]$. This proves (ii) in the case of SKInT-reduction. (Notice that we have in fact shown that the standard normalization sequence for u terminates. This proof would also work with $[\![u]\!](x_0, \dots, x_{n-1})$ instead of $|[\![u]\!]_\oplus(x_0, \dots, x_{n-1})|$.)

For SKInT$_\eta$-reduction, notice that every SKInT-normalizable term is also SKInT$_\eta$-normalizable. Indeed, observe that if $u \longrightarrow v$ by the η-rule (ηS_ℓ), and u is SKInT-normal, then so is v (see Appendix I, Claims (d) and (e)) . This finishes to prove (ii). $\Box$

The Normalization Theorem has an important corollary. Recall that a translation from λ-terms to a given language (say, SKInT) *preserves* strong normalization (resp. weak normalization, solvability) if and only if the translation of every strongly normalizing λ-term (resp. weakly normalizing, resp. solvable) is strongly normalizing (resp. weakly normalizing, resp. solvable).

Corollary 6 (Preservation of Normalization Properties) *Every translation mapping $\mathcal{S}$-typable λ-terms to S-typable* SKInT*-terms preserves strong normalization. Every translation mapping $\mathcal{S}\omega$-typable λ-terms to $S\omega$-typable* SKInT*-terms preserves solvability. Every translation mapping λ-terms having a definite positive typing in $\mathcal{S}\omega$ to* SKInT*-terms having a definite positive typing in $S\omega$ preserves weak normalization.*

Proof: Notice that these translations need not map λ-terms to SKInT-terms of the same type: we just need to preserve typability, not the types themselves. Every strongly normalizing (resp. weakly normalizing, solvable) λ-term is typable in system $\mathcal{S}$ (resp. in $\mathcal{S}\omega$, in $\mathcal{S}\omega$ with a definite positive typing) [Say97]; the result then follows from Theorem 5. $\Box$

It follows that the translations L^* and H^* of [GGL98] each preserve strong normalization, weak normalization and solvability. This works also in the presence of η-rules. Also, Corollary 6 is more general: essentially, any reasonable translation from the λ-calculus to SKInT will preserve all three normalization properties.

Corollary 6 depends on the fact that strongly normalizing, resp. weakly normalizing, resp. solvable terms in the λ-calculus are all characterized in terms of types. We end this section by showing that the same holds in SKInT.

First, define S, resp. *$S\omega$-type substitutions* θ as finite maps from type variables, a.k.a. base types in B, to S, resp. $S\omega$-types. For any type or context a, $a\theta$ denotes the result of applying θ to a; $[\tau/b]$ denotes the substitution mapping b to τ. The following is easy:

Lemma 7 *If $\Gamma \vdash u : \tau$ is derivable in S, then for every S-type substitution θ, $\Gamma\theta \vdash u : \tau\theta$ is derivable in S.*

If $\Gamma \vdash u : \tau$ is derivable in $S\omega$, then for every $S\omega$-type substitution θ, $\Gamma\theta \vdash u : \tau\theta$ is derivable in $S\omega$.

If $\Gamma^- \vdash u : \tau^+$ is definite positive and derivable in $S\omega$, then for every S-type substitution θ, $\Gamma^-\theta \vdash u : \tau^+\theta$ is definite positive and derivable in $S\omega$.

Lemma 8 *Every* SKInT*-normal term u has a typing in system S. Every spine-normal* SKInT*-term u has a typing in $S\omega$.*

Proof: We first observe that every type can be written uniquely $\mu_0 \to \ldots \to \mu_{n-1} \to b$, where $b \in B$. We call n the *arity* of the type. By extension, we call arity of a typing $\Gamma \vdash u : \tau$ the arity of τ. We call a typing as above *S-normal* if $n \geq 1$ and $\mu_{n-1} = [b']$ with b' a base type other than b; we call it *$S\omega$-normal* if $n \geq 1$ and $\mu_{n-1} = \omega$.

Let the *degree* $d(u)$ of a SKInT-term u be defined by: $d(x) =_{\mathrm{df}} 0$, $d(I_\ell) =_{\mathrm{df}} \ell + 1$, $d(S_\ell(v, w)) =_{\mathrm{df}} \ell$, $d(K_\ell(v)) =_{\mathrm{df}} \ell + 1$. We show the more general claims that: (i) every SKInT-normal term u has a normal S-typing of arity $d(u) + 1$, and: (ii) every spine-normal term u has a normal $S\omega$-typing of arity $d(u) + 1$. This is by structural induction on u.

Notice first that: $(*)$ whenever a term u has an S-normal, resp. $S\omega$-normal typing, then it also has S-normal, resp. $S\omega$-normal typings $\Gamma \vdash u : \tau$ of arbitrary higher arities in the same system: indeed, if u has a normal typing $\Gamma \vdash u : \tau$ of arity n as above (with b the base type at the end), then it also has a normal typing of arity $n+1$, namely $\Gamma\theta \vdash u : \tau\theta$, by Lemma 7, with $\theta =_{\mathrm{df}} [[b''] \to b'/b]$, $b'' \neq b'$ in the case of system S, $\theta =_{\mathrm{df}} [\omega \to b'/b]$ in the case of $S\omega$. Claim $(*)$ then follows by an easy induction on n.

If u is a variable, then the normal typing $x : [b'] \to b \vdash x : [b'] \to b$ establishes (i), and $x : \omega \to b \vdash x : \omega \to b$ establishes (ii). If u is of the form I_ℓ, then we can choose the typing $\vdash I_\ell : \mu_0 \to \ldots \to \mu_{\ell-1} \to [[b'] \to b] \to [b'] \to b$ with $\mu_0, \ldots, \mu_{\ell-1}$ any S-types and $b' \neq b$ for (i), and $\vdash I_\ell : \mu_0 \to \ldots \to \mu_{\ell-1} \to [\omega \to b] \to \omega \to b$ for (ii).

When u is of the form $S_\ell(v,w)$, then, first, define the conjunction $\Gamma' \wedge \Gamma''$ of two contexts Γ' and Γ'' as the collection of bindings $x : \mu' \wedge \mu''$ (when $x : \mu' \in \Gamma'$ and $x : \mu'' \in \Gamma''$), $x : \mu'$ (if $x : \mu' \in \Gamma'$ but x does not appear in Γ''), and $x : \mu''$ (if $x : \mu'' \in \Gamma''$ but x does not appear in Γ''). Notice also that (by examination of the reduction rules), if u is spine-normal (in particular, normal), then $d(v) \leq \ell$. We show claim (i) as follows: by induction, v has an S-normal typing of arity $d(v)+1 \leq \ell+1$, hence by $(*)$ it has an S-normal typing of arity exactly $\ell+1$, say $\Gamma' \vdash v : \mu'_0 \to \ldots \to \mu'_{\ell-1} \to [b'] \to b$; by induction hypothesis again, w has an S-typing, hence by $(*)$ we may assume w.l.o.g. that w has an S-typing of arity at least ℓ, say $\Gamma' \vdash w : \mu''_0 \to \ldots \to \mu''_{\ell-1} \to \tau''$; then, we can derive $\Gamma'[\tau''/b'] \vdash v : \mu_0 \to \ldots \to \mu_{\ell-1} \to [\tau''] \to b$, where $\mu_i =_{\mathrm{df}} \mu'_i[\tau''/b']$, $0 \leq i < n$, by Lemma 7; then $\Gamma'[\tau''/b'] \wedge \Gamma'' \vdash u : \mu_0 \to \ldots \to \mu_{\ell-1} \to b$ is an S-typing of arity $d(u) = \ell$; substituting b for, say, $[b''] \to b'''$ (using Lemma 7), we get the required S-normal typing of arity $d(u)+1$. Showing (ii) is easier: by induction, v has an $S\omega$-normal typing of arity $d(v)+1 \leq \ell+1$, hence by $(*)$ it has an $S\omega$-normal typing of arity $\ell+1$, say $\Gamma' \vdash v : \mu'_0 \to \ldots \to \mu'_{\ell-1} \to \omega \to b$; then $\Gamma \vdash u : \mu_0 \to \ldots \to \mu_{\ell-1} \to \omega \to b'$ is the required $S\omega$-normal typing of arity $\ell+1$, where $\Gamma =_{\mathrm{df}} \Gamma'[\omega \to b'/b]$, and $\mu_i =_{\mathrm{df}} \mu'_i[\omega \to b'/b]$, $0 \leq i < \ell$.

When u is of the form $K_\ell(v)$, then observe that (by examination of the reduction rules), if u is spine-normal (or normal), then $d(v) \leq \ell$. We show (i) as follows: by induction v has an S-normal typing of arity $d(v)+1$, hence by $(*)$ an S-normal typing $\Gamma' \vdash v : \mu'_0 \to \ldots \to \mu'_{\ell-1} \to [b'] \to b$ of arity $\ell+1$; then $\Gamma' \vdash K_\ell(v) : \mu'_0 \to \ldots \to \mu'_{\ell-1} \to \mu \to [b'] \to b$ is the required S-normal typing for u, for any S-type μ. And (ii) follows by a similar construction, replacing $[b']$ by ω and letting μ be any $S\omega$-type. □

Contrarily to what happens in the λ-calculus, types in $S\omega$ are not preserved by the inverse of SKInT-reduction. However:

Lemma 9 *If $u \longrightarrow^* v$ in* SKInT_η*, and $\Gamma \vdash v : \tau$ in $S\omega$, then $\Gamma\theta \vdash u : \tau\theta$ for some S-type substitution θ.*

Proof: More concisely, we shall say that whenever v has some $S\omega$-type τ, then u has type $\tau\theta$ for some S-substitution θ (where the contexts Γ and $\Gamma\theta$ are understood). We first show this when $u = l$, $v = r$, and $l \to r$ is any of the reduction rules. There are only three interesting rules:

- (SI_ℓ): $l = S_\ell(I_\ell, w)$, $r = w$. If τ has arity $n \geq \ell$, then τ is of the form $\mu_0 \to \ldots \to \mu_{\ell-1} \to \tau'$, and we can take I_ℓ of type $\mu_0 \to \ldots \to \mu_{\ell-1} \to [\tau'] \to \tau'$, so that l has type τ. If τ has arity $n < \ell$, then let τ be $\mu_0 \to \ldots \to \mu_{n-1} \to b$, and θ be $[\tau'/b]$, where τ' is any S-type of arity at least $\ell - n$: by Lemma 7, r has type $\tau\theta$, and since $\tau\theta$ has arity at least ℓ, then as above l has type $\tau\theta$.

- (SK_ℓ): $l = S_\ell(K_\ell(u), w)$, $r = u$. If τ has arity $n \geq \ell+1$, then write τ as $\mu_0 \to \ldots \to \mu_{\ell-1} \to \mu_\ell \to \tau'$: we can give $K_\ell(u)$ the $S\omega$-type $\mu_0 \to \ldots \to \mu_{\ell-1} \to \omega \to \mu_\ell \to \tau'$, and therefore l has type τ as well (notice that we need not give w a type). Otherwise, as in the (SI_ℓ) case, r and l have type $\tau\theta$ for some S-substitution θ of arity $\ell + 1 - n$.

- (ηS_ℓ): $l = S_{\ell+1}(K_\ell(u), I_\ell)$, $r = u$. If the type τ of r has arity $n \geq \ell+1$, then write τ as $\mu_0 \to \ldots \to \mu_\ell \to \tau'$, and μ_ℓ as $[\tau_1, \ldots, \tau_k]$. We can give I_ℓ all the types $\mu_0 \to \ldots \to \mu_{\ell-1} \to \mu_\ell \to \tau_i$, $1 \leq i \leq k$, and we can give $K_\ell(u)$ the type $\mu_0 \to \ldots \to \mu_{\ell-1} \to \mu_\ell \to [\tau_1, \ldots, \tau_k] \to \tau'$, so l can be given type τ again. If $n < \ell+1$, then, if τ is of the form $\mu_0 \to \ldots \to \mu_{n-1} \to b$, let θ be $[\tau'/b]$ for any S-type τ' of arity at least $\ell + 1 - n$: then l has type $\tau\theta$.

For all the other rules, we can choose the identity substitution for θ (see Appendix H).

We now claim that whenever $u \longrightarrow v$—namely, when $u = \mathcal{C}[l]$, $v = \mathcal{C}[r]$, and $l \to r$ is some reduction rule—and v has type τ in $S\omega$, then u has some type $\tau\theta$ in $S\omega$, where θ is an S-substitution. This is a straightforward structural induction on $\mathcal{C}$, using Lemma 7.

The Lemma then follows by induction on the length of the reduction $u \longrightarrow^* v$. □

It follows:

Theorem 10 *Every solvable* SKInT*-term has an $S\omega$-typing. Every* SKInT*-weakly normalizing, resp.* SKInT_η*-weakly normalizing term has a definite positive $S\omega$-typing.*

Proof: Let u be solvable. Then $u \longrightarrow^{s*} v$, where v is spine-normal; by Lemma 8, v has an $S\omega$-typing $\Gamma \vdash v : \tau$; by Lemma 9, u has a typing of the form $\Gamma\theta \vdash u : \tau\theta$, where θ is an S-substitution. This is clearly an $S\omega$-typing.

On the other hand, if u is weakly normalizing (in SKInT or in SKInT$_\eta$), then $u \longrightarrow^* v$ for some SKInT-normal term v; by Lemma 8, v has a definite positive $S\omega$-typing $\Gamma^- \vdash v : \tau^+$; by Lemma 9, u has a typing of the form $\Gamma^-\theta \vdash u : \tau^+\theta$, where θ is an S-substitution. By Lemma 7, this typing is therefore not only an $S\omega$-typing, but is also definite positive. □

Theorem 11 *If u is strongly normalizing in* SKInT, *resp.* SKInT$_\eta$, *then it has an S-typing.*

Proof: As u is strongly normalizing, any normalization strategy terminates. Choose any *innermost* strategy, i.e. any strategy that reduces only redexes whose strict subterms are all normal. (In particular, the redex $S_\ell(K_\ell(u_1), u_2)$ can only be reduced when u_1 and $K_\ell(u_1)$ are normal.) Let $\nu(u)$ denote the length of the longest reduction sequence in SKInT starting from u according to this strategy.

We show the claim by induction on $\nu(u)$. If $\nu(u) = 0$, then this is by Lemma 8. Otherwise, assume that $u \longrightarrow v$ (so that $\nu(v) < \nu(u)$, hence by induction v has an S-typing $\Gamma \vdash v : \tau$). If the reduction from u to v is by any rule except (SK_ℓ), $\ell \geq 0$, then u has an S-typing $\Gamma\theta \vdash u : \tau\theta$, where θ is an S-substitution: this is as in the proof of Lemma 9.

In case $u \longrightarrow v$ by (SK_ℓ), this does not work any longer, since we cannot use ω (not an S-type). Instead, observe that u can be written as $\mathcal{C}[S_\ell(K_\ell(u_1), u_2)]$, and that $v = \mathcal{C}[u_1]$. By induction hypothesis, and since $\nu(u_2) < \nu(u)$, u_2 has an S-typing $\Gamma_2 \vdash u_2 : \tau_2$, and w.l.o.g. we may assume that τ_2 has arity at least ℓ, i.e. that $\tau_2 = \mu''_0 \to \ldots \to \mu''_{\ell-1} \to \tau''$. Since the chosen reduction strategy is innermost, u_1 is normal, and by Lemma 8 (more precisely, by Claim (i) in its proof), u_1 has an S-normal typing of degree exactly $d(u_1) + 1$; but since the reduction is innermost again, $K_\ell(u_1)$ is normal as well, so $d(u_1) \leq \ell$, and (by Remark $(*)$ in the proof of Lemma 8) therefore u_1 has an S-normal typing of arity $\ell{+}1$, say $\Gamma_1 \vdash u_1 : \mu'_0 \to \ldots \to \mu'_{\ell-1} \to [b'] \to b$. So we may now derive $\Gamma \vdash S_\ell(K_\ell(u_1), u_2) : \mu_0 \to \ldots \to \mu_{\ell-1} \to b$, where $\Gamma =_{\mathrm{df}} \Gamma_1[\tau''/b'] \wedge \Gamma_2$, $\mu_i =_{\mathrm{df}} \mu'_i[\tau''/b'] \wedge \mu''_i$ for each i, $0 \leq i < \ell$. It follows that u itself has an S-typing (which is an instance of the latter), by a straightforward induction on $\mathcal{C}$.

The case of SKInT$_\eta$-strongly normalizing terms is completely similar. (Or observe that any SKInT$_\eta$-strongly normalizing term is SKInT-strongly normalizing, by postponement of η.) □

It follows finally that normalization properties can be characterized in terms of typability, both in SKInT and in SKInT$_\eta$, just as in the λ-calculus:

Corollary 12 *The following equivalences hold, for any* SKInT*-term u:*

u solvable $\Leftrightarrow$ u typable in $S\omega$

u SKInT-weakly normalizing $\Leftrightarrow$ u SKInT$_\eta$-weakly normalizing $\Leftrightarrow$ u has a definite positive $S\omega$-typing

u SKInT-strongly normalizing $\Leftrightarrow$ u SKInT$_\eta$-strongly normalizing $\Leftrightarrow$ u typable in S.

A A Few Derived Rules in the System of Figure 5

First, observe that weakening is admissible, that is, whenever $\Gamma \vdash t : \tau$ is derivable in the system of Figure 5, then so is $\Gamma, \Delta \vdash t : \tau$ for any Δ (whose domain is disjoint from that of Γ). Indeed, just add Δ to the left of each sequent from the bottom of the derivation up to the points where we reach axioms (the variable rule) or the abstraction rule.

We now show that the following rule:

$$\frac{\Gamma \vdash t : \tau \quad \Gamma, x : \tau \vdash s : \tau'}{\Gamma \vdash s[t/x] : \tau'}\ (Cut)$$

is admissible as well. This is by structural induction on s. If s is x, then $\tau = \tau'$ and the result is clear. If s is another variable y, then $y : \tau'$ occurs in Γ, and $s[t/x] = y$, so the result is clear again. When s is an application

$s_1 s_2$, the result follows directly by induction hypothesis. When s is a closure $\langle x' \cdot s', \{x_1 := s_1, \ldots, x_k := s_k\}\rangle$, typed as follows:

$$\frac{\Gamma, x : \tau \vdash s_1 : \tau_1' \to \tau_1'' \quad \ldots \quad \Gamma, x : \tau \vdash s_k : \tau_k' \to \tau_k'' \qquad x_1 : \tau_1' \to \tau_1'', \ldots, x_k : \tau_k' \to \tau_k'', x' : \tau'' \vdash s' : \tau'''}{\Gamma, x : \tau \vdash \langle x' \cdot s', \{x_1 := s_1, \ldots, x_k := s_k\}\rangle : \tau'' \to \tau'''}$$

so that $\tau' = \tau'' \to \tau'''$, then by induction hypothesis we have derivations of $\Gamma \vdash s_i[t/x] : \tau_i' \to \tau_i''$, $1 \leq i \leq k$. Therefore, by the abstraction rule again, we can derive $\Gamma \vdash \langle x' \cdot s', \{x_1 := s_1[t/x], \ldots, x_k := s_k[t/x]\}\rangle : \tau'' \to \tau'''$, i.e. $\Gamma \vdash s[t/x] : \tau'$.

B Proof of Theorem 1

Only if direction (soundness): by structural induction on t. The only non-trivial case is for closures $\langle x \cdot s, \{x_1 := t_1, \ldots, x_k := t_k\}\rangle$, which we assume typed as shown in Figure 5. Fix the frame, and consider any arbitrary world w satisfying Γ (i.e., such that $w \models \tau$ for every $x : \tau$ in Γ). By induction hypothesis, $w \models \tau_i' \to \tau_i''$ for every i, $1 \leq i \leq k$. It follows that $w' \models \tau_i' \to \tau_i''$ for any w' such that $w \leq w'$: indeed, the set of worlds at which an arrow type holds is clearly upwards-closed. Then by induction hypothesis again, for every w' such that $w \leq w'$ and $w' \models \tau'$, then $w' \models \tau''$. By definition, this means that $w \models \tau' \to \tau''$. Now w was assumed arbitrary satisfying Γ, and this concludes the proof.

If direction (completeness): assume that there is no λ_{clos}-term t such that $\Gamma_0 \vdash t : \tau_0$ is derivable. We shall build a frame and a world w_0 such that w_0 satisfies Γ_0 but $w_0 \not\models \tau_0$.

We shall consider *signed formulas* F, which are either positive $+\tau$ or negative $-\tau$, where τ is a type. (Intuitively, $+\tau$ means that τ is assumed true, and $-\tau$ means that τ is assumed false.) A set S of signed formulas is *consistent* if and only if for every finite set of positive formulas $+\tau_1, \ldots, +\tau_n$ (not necessarily distinct) and every negative formula $-\tau$ in S, there is no λ_{clos}-term t such that $x_1 : \tau_1, \ldots, x_n : \tau_n \vdash t : \tau$. (Intuitively, this means that it is not contradictory to assume $\tau_1, \ldots, \tau_n$ true and τ false—hence the name "consistent".) Call S *maximally consistent* if and only if S is consistent but no strict superset of S is consistent.

We first claim that: (i) every consistent set is included in some maximally consistent set. Indeed, apply Zorn's Lemma to the set of all consistent sets ordered by $\subseteq$. To apply it, we just have to check that, given a linearly ordered set of consistent sets $(S_i)_{i \in I}$, $\bigcup_{i \in I} S_i$ is itself consistent. Indeed, assume on the contrary that $\bigcup_{i \in I} S_i$ is inconsistent. By definition, there must be positive formulas $+\tau_1, \ldots, +\tau_n$ and a negative formula $-\tau$ in $\bigcup_{i \in I} S_i$, and a λ_{clos}-term t such that $x_1 : \tau_1, \ldots, x_n : \tau_n \vdash t : \tau$ is derivable. Since the mentioned signed formulas are only finitely many, they are all in some finite union of S_i's, and since $(S_i)_{i \in I}$ is linearly ordered, they all belong to some S_i; but then, this S_i is inconsistent by definition, which is absurd. This proves (i).

We then claim that: (ii) given any maximally consistent set S, for every type τ, either $+\tau$ or $-\tau$ (but not both) belong to S. Of course, they cannot both belong to S, since $x : \tau \vdash x : \tau$ is derivable, and S is consistent. On the other hand, assume that neither $+\tau$ nor $-\tau$ is in S. Since S is maximal, $S \cup \{+\tau\}$ and $S \cup \{-\tau\}$ are inconsistent. Therefore, on the one hand there are positive formulas $+\tau_1, \ldots, +\tau_n$ and a negative formula $-\tau'$ in S, and a λ_{clos}-term t such that $x_1 : \tau_1, \ldots, x_n : \tau_n, y_1 : \tau, \ldots, y_p : \tau \vdash t : \tau'$ is derivable (and $p \geq 1$, otherwise S would be inconsistent); and on the other hand, there are positive formulas $\tau_1', \ldots, \tau_m'$ in S and a λ_{clos}-term t' such that $x_1' : \tau_1', \ldots, x_m' : \tau_m' \vdash t' : \tau$ is derivable (the type on the right must be τ, otherwise it would appear negated in S, and S would be inconsistent). We may assume w.l.o.g. that $x_1, \ldots, x_n, y_1, \ldots, y_p, x_1', \ldots, x_m'$ are pairwise distinct. Then, a simple structural induction on t shows that $x_1 : \tau_1, \ldots, x_n : \tau_n, x_1' : \tau_1', \ldots, x_m' : \tau_m' \vdash t[t'/y_1, \ldots, t'/y_p] : \tau'$ is derivable. But $+\tau_1, \ldots, +\tau_n, +\tau_1', \ldots, +\tau_m'$ and $-\tau'$ are in S by assumption, showing that S is inconsistent, which is absurd. This proves (ii).

It follows that: (iii) every maximally consistent set S is closed under deduction, in the following sense: if $x_1 : \tau_1, \ldots, x_n : \tau_n \vdash t : \tau$ is derivable and $+\tau_1, \ldots, +\tau_n$ belong to S, then so does $+\tau$. Indeed, by (ii), otherwise $-\tau$ would be in S, which would imply that S is inconsistent.

Now define $\mathcal{W}$ as the set of all maximally consistent sets w of signed formulas. Let $\leq$ be defined by $w \leq w'$ if and only if, for every positive formula of the form $+(\tau_1 \to \tau_2)$ in w such that $+\tau_1$ is in w', then $+\tau_2$ is in w'. Finally, define ρ as mapping every base type b to the set of worlds w such that $+b \in w$.

We claim that: (iv) $(\mathcal{W}, \leq, \rho)$ is a near-intuitionistic frame, i.e., that $\leq$ is reflexive and transitive. Reflexivity: for every world, i.e. maximally consistent set w containing $+(\tau_1 \to \tau_2)$ and $+\tau_1$, notice that $x_1 : \tau_1 \to \tau_2, x_2 : \tau_1 \vdash x_1 x_2 : \tau_2$ is derivable, so by (iii) $+\tau_2 \in w$; as this holds for every τ_1 and τ_2, by definition $w \leq w$. Transitivity: assume $w \leq w'$ and $w' \leq w''$; then, whenever $+(\tau_1 \to \tau_2)$ is in w and $+\tau_1$ is in w'', we want to show that $+\tau_2$ in in w''. Now, fix a type τ, and let $\top$ be $\tau \to \tau$. The judgment $\vdash \langle z \cdot z \rangle : \top$ is derivable, so $+\top$ belongs to every world by (iii), in particular to w'. On the other hand, $x_1 : \tau_1 \to \tau_2 \vdash \langle y \cdot x_1 \rangle : \top \to \tau_1 \to \tau_2$ is also derivable, so by (iii) and since $+(\tau_1 \to \tau_2) \in w$, it follows that $+(\top \to \tau_1 \to \tau_2) \in w$. By definition of $\leq$, since $w \leq w'$ and $+\top \in w'$, it follows that $+(\tau_1 \to \tau_2) \in w'$. By definition of $\leq$ again, since $w' \leq w''$ and $+\tau_1 \in w''$, it follows that $+\tau_2 \in w''$. Since τ_1 and τ_2 are arbitrary, it follows that $w \leq w''$.

We now claim that: (v) for every type τ, for every world w, $w \models \tau$ if and only if $+\tau \in w$. This is by structural induction on τ. This is clear when τ is a base type (by definition of ρ). So let τ be $\tau' \to \tau''$:

- (If direction:) if $+(\tau' \to \tau'') \in w$, then by definition of $\leq$, for every w' such that $w \leq w'$ and $+\tau' \in w'$, then $+\tau'' \in w'$. By induction hypothesis, this is equivalent to: for every w' such that $w \leq w'$ and $w' \models \tau'$, then $w \models \tau''$. That is, $w \models \tau' \to \tau''$, by definition of $\models$.

- (Only if direction:) Assume $w \models \tau' \to \tau''$ but $+(\tau' \to \tau'') \notin w$, by contradiction. Let $w^{\to}$ be the set of all positive arrow types in w; let S be $w^{\to} \cup \{+\tau', -\tau''\}$.

  Observe that S is consistent: otherwise there would be elements $+\tau_1, \ldots, +\tau_n$ of $w^{\to}$ and a λ_{clos}-term t such that $x_1 : \tau_1, \ldots, x_n : \tau_n, y_1 : \tau', \ldots, y_k : \tau' \vdash t : \tau''$ is derivable; w.l.o.g, we can assume that $k = 1$ (if $k = 0$, notice that weakening is admissible, and we may add the dummy assumption $y_1 : \tau'$ on the left; otherwise, replace t by $t[y_1/y_2, \ldots, y_1/y_k]$); then, since $\tau_1, \ldots, \tau_n$ are in $w^{\to}$, they are arrow types, so the abstraction rule is applicable, and $x_1 : \tau_1, \ldots, x_n : \tau_n \vdash \langle y_1 \cdot t \rangle : \tau' \to \tau''$ is derivable; by (iii) and since by assumption $+\tau_1, \ldots, +\tau_n$ are in w, this implies that $+(\tau' \to \tau'')$ is in w, which contradicts our assumption that $+(\tau' \to \tau'') \notin w$.

  Since S is consistent, by (i) there is a maximally consistent set w' containing S. We claim that $w \leq w'$. Indeed, for every $+(\tau_1 \to \tau_2)$ in w, $+(\tau_1 \to \tau_2)$ is in $w^{\to}$, hence in S, and therefore in w'. So, whenever $+\tau_1$ is in w', by (iii) $+\tau_2$ is in w' as well: since τ_1 and τ_2 are arbitrary, $w \leq w'$.

  Since $w \models \tau' \to \tau''$ (by assumption) and $w \leq w'$, $w' \models \tau'$ implies $w' \models \tau''$. But by construction $+\tau'$ is in S, hence in w', so by induction $w' \models \tau'$, and therefore $w' \models \tau''$. By induction again, we must have $+\tau'' \in w'$. But by construction $-\tau''$ is in S, hence in w'. Since both $+\tau''$ and $-\tau''$ are in w', it follows that w' is inconsistent, which is absurd.

Now recall that by assumption, there is no λ_{clos}-term t such that $\Gamma_0 \vdash t : \tau_0$ is derivable. In other words, letting Γ_0 be $x_1 : \tau_1, \ldots, x_n : \tau_n$, the set $\{+\tau_1, \ldots, +\tau_n, -\tau_0\}$ is consistent. By (i), there is a maximally consistent set w_0 containing it. By (v), $w_0 \models \tau_1$ and $\ldots$ and $w_0 \models \tau_n$. Moreover, $w_0 \not\models \tau_0$, otherwise by (v) again $+\tau_0$ would be in w_0, but this would contradict the consistency of w_0, since $-\tau_0$ is already in w_0. This terminates the construction of a frame $(\mathcal{W}, \leq, \rho)$ and a world $w_0 \in \mathcal{W}$ such that w_0 satisfies Γ_0 but not w_0.

C Proof of Theorem 2

First, the equivalence between the systems of Figure 5 and 6 is obtained as follows. We translate the rules of Figure 5 as the following combinations of sequent rules: the axiom $\Gamma, x : \tau \vdash x : \tau$ is the same in both calculi, application becomes:

$$\dfrac{\Gamma \vdash s : \tau_1 \to \tau_2 \quad \dfrac{\dfrac{}{\Gamma, y : \tau_2 \vdash y : \tau_2} \quad \Gamma \vdash t : \tau_1}{\Gamma, x : \tau_1 \to \tau_2 \vdash xt : \tau_2}}{\Gamma \vdash st : \tau_2}\ (Cut)$$

and abstraction becomes:

$$\dfrac{\Gamma \vdash t_1 : \tau_1' \to \tau_1'' \ldots \Gamma \vdash t_k : \tau_k' \to \tau_k'' \qquad \dfrac{x_1 : \tau_1' \to \tau_1'', \ldots, x_k : \tau_k' \to \tau_k'', x : \tau' \vdash s : \tau''}{x_1 : \tau_1' \to \tau_1'', \ldots, x_k : \tau_k' \to \tau_k'' \vdash \langle x \cdot s, \{x_1 := x_1, \ldots, x_k := x_k\}\rangle : \tau''}}{\Gamma \vdash \langle x \cdot s, \{x_1 := t_1, \ldots, x_k := t_k\}\rangle : \tau''}$$

where the last line follows from its premises by uses of weakenings (which is admissible in the Gentzen system of Figure 6) and of (Cut), used k times.

Conversely, we translate the rules of the Gentzen system into natural deduction rules as follows. The axiom is the same in both systems. We have already seen that cut was admissible in the natural deduction system (see Appendix A). The left implication rule becomes:

$$\dfrac{\dfrac{\overline{\Gamma, x : \tau_1 \to \tau_2 \vdash x : \tau_1 \to \tau_2} \qquad \Gamma \vdash s : \tau_1}{\Gamma, x : \tau_1 \to \tau_2 \vdash xs : \tau_2} \qquad \Gamma, y : \tau_2 \vdash t : \tau}{\Gamma, x : \tau_1 \to \tau_2 \vdash t[xs/y] : \tau}\ (Cut)$$

where some weakenings (which are admissible in the natural deduction system) have been silently applied, and (Cut) is the admissible cut rule in the natural deduction system. The right application rule of the sequent calculus is translated as:

$$\dfrac{\overline{\Gamma \vdash x_1 : \tau_1' \to \tau_1''} \ \ldots\ \overline{\Gamma \vdash x_k : \tau_1' \to \tau_k''} \qquad \Gamma, x : \tau' \vdash s : \tau''}{\Gamma \vdash \langle x \cdot s, \{x_1 := x_1, \ldots, x_k := x_k\}\rangle : \tau' \to \tau''}$$

where Γ (the arrowed context) is written $x_1 : \tau_1' \to \tau_1'', \ldots, x_k : \tau_k' \to \tau_k''$.

We now show the second part of the Theorem, namely that (Cut) is eliminable from the system of Figure 6.

Subject Reduction: First, we claim that subject reduction (for the natural deduction system of Figure 5) holds, namely: if $\Gamma \vdash s : \tau$ is derivable in the natural deduction system, and $s \longrightarrow t$ in λ_{clos}, then $\Gamma \vdash t : \tau$ is derivable as well. This is a tedious but straightforward check, which we leave to the reader.

Strong Normalization: Now, if s is a simply typed λ_{clos}-term, i.e. if $\Gamma \vdash s : \tau$ is derivable in the system of Figure 5, then s is strongly normalizing, that is, there is no infinite reduction starting from s. There are many ways to prove this. One is to use the $s \mapsto s^*$ translation and use the fact that the simply-typed SKInT-calculus is strongly normalizing. Or we may translate λ_{clos} to λ_{S4} [GL96], by:

$$\begin{aligned} g(x) &= x \\ g(st) &= (\mathsf{unbox}\ g(s))g(t) \\ g(\langle x \cdot s, \{x_1 := t_1, \ldots, x_k := t_k\}\rangle) &= \mathsf{box}\ \lambda x \cdot g(s)\ \mathsf{with}\ g(t_1), \ldots, g(t_k)\ \mathsf{for}\ x_1, \ldots, x_k \end{aligned}$$

and also translate types by $g(b) = b$ for all base types b, and $g(\tau_1 \to \tau_2) = \Box(g(\tau_1) \Rightarrow g(\tau_2))$; we then use the fact that the simply typed λ_{S4}-calculus is strongly normalizing.

A more direct way is to translate the λ_{clos}-terms to λ-terms by making all substitutions implicit, namely:

$$\begin{aligned} h(x) &=_{\text{df}} x \\ h(st) &=_{\text{df}} h(s)h(t) \\ h(\langle x \cdot s, \{x_1 := t_1, \ldots, x_k := t_k\}\rangle) &=_{\text{df}} \lambda x \cdot s[h(t_1)/x_1, \ldots, h(t_k)/x_k] \end{aligned}$$

Observe that if $\Gamma \vdash t : \tau$ is derivable in the system of Figure 5, then $\Gamma \vdash h(t) : \tau$ in the simply-typed λ-calculus. Since every λ-term contains only finitely many redexes, by König's Lemma every strongly normalizing (in particular, every simply-typed) λ-term has a longest reduction sequence. Let $\nu(t)$ denote the length of the longest reduction sequence of the strongly normalizing term t.

Observe, then, that: (i) if $s \longrightarrow^* t$ in λ_{clos}, then $h(s)$ rewrites to $h(t)$ in the λ-calculus. This is a trivial induction on the length of reductions, then on the structure of terms.

Observe also (as above) that, if s is a strongly normalizing λ_{clos}-term, then we can define the length $\nu_{clos}(s)$ of the longest λ_{clos}-reduction starting from s, and also the sum $\sigma(s)$ of the lengths of all reductions starting from s.

Let n be any integer, and assume that: $(*)$ for every λ_{clos}-term s such that $\nu(h(s')) < n$ for every subterm s' of s, s is strongly normalizing. We claim that:

(ii) if s and t are strongly normalizing λ_{clos}-terms and $\nu(h(st)) < n+1$, then st is a strongly normalizing λ_{clos}-term. This is by induction on $\nu_{clos}(s) + \nu_{clos}(t)$. Consider any reduction R starting from st. If R is empty, then the result is clear. Otherwise, consider the first redex that R contracts in st: if st rewrites to $s't$, where $s \longrightarrow s'$, then by *(i)* $h(st)$ rewrites to $h(s't)$ in the λ-calculus (so $\nu(h(s't)) < n+1$), and $\nu_{clos}(s) + \nu_{clos}(t) > \nu_{clos}(s') + \nu_{clos}(t)$, so the induction hypothesis applies, and therefore R is finite. Similarly if the first rewrite occurs in t. Finally, if the first rewrite in R occurs at the top of st, then this must be by rule (β), so s is of the form $\langle x \cdot s', \{x_1 := t_1, \ldots, x_k := t_k\}\rangle$, and st rewrites to $s'[t/x, t_1/x_1, \ldots, t_k/x_k]$: then $h(st) = (\lambda x \cdot h(s')[h(t_1)/x_1, \ldots, h(t_k)/x_k])h(t) \longrightarrow h(s')[h(t)/x, h(t_1)/x_1, \ldots, h(t_k)/x_k]$ (by the variable naming condition) $= h(s'[t/x, t_1/x_1, \ldots, t_k/x_k])$ (straightforward structural induction on s'). In particular, $\nu(h(s'[t/x, t_1/x_1, \ldots, t_k/x_k])) < \nu(h(st))$, so $\nu(h(s'[t/x, t_1/x_1, \ldots, t_k/x_k])) < n$, and therefore R is finite by assumption $(*)$.

(iii) if s, t_1, $\ldots$, t_k are strongly normalizing λ_{clos}-terms, then $\langle x \cdot s, \{x_1 := t_1, \ldots, x_k := t_k\}\rangle$ is strongly normalizing. This is by induction on $(\sigma(s) + \sigma(t_1) + \ldots + \sigma(t_k), |t_1| + \ldots + |t_k|)$ ordered lexicographically, where the size $|t|$ of a term is defined by: $|x| =_{\text{df}} 1$, $|st| =_{\text{df}} |s| + |t| + 1$, $|\langle x' \cdot s', \{x'_1 := t'_1, \ldots, x'_p := t'_p\}\rangle| =_{\text{df}} |s'| + |t'_1| + \ldots + |t'_p| + 1$. If $\langle x \cdot s, \{x_1 := t_1, \ldots, x_k := t_k\}\rangle$ is normal, the result is trivial. Otherwise, consider the first reduction step: if it occurs in s, t_1, $\ldots$, or t_k, then $\langle x \cdot s, \{x_1 := t_1, \ldots, x_k := t_k\}\rangle$ rewrites to a term of the same form, but with a decreased $\sigma(s) + \sigma(t_1) + \ldots + \sigma(t_k))$, so that the induction applies, and therefore the reduction terminates. Finally, if the first reduction step occurs at the top, this may be by rule (ι), (0) or (2); then $\sigma(s) + \sigma(t_1) + \ldots + \sigma(t_k))$ does not increase, but the $|t_1| + \ldots + |t_k|$ decreases in each case: again, the induction hypothesis applies, so the reduction terminates. This proves *(iii)*.

So, under the assumption $(*)$, every λ_{clos}-term s such that $\nu(h(s')) < n+1$ for every subterm s' of s is strongly normalizing: indeed, this is by structural induction on s, using *(ii)* in the case of applications and *(iii)* in the case of closures (the case of variables is trivial).

By induction on n, it follows that for every λ_{clos}-term s such that $h(s')$ is λ-strongly normalizing for every subterm s' of s, s is λ_{clos}-strongly normalizing. In particular, every simply-typed λ_{clos}-term is strongly normalizing. (We use the fact that if a λ_{clos}-term is typable, then all its subterms are as well.)

Notice that we have not just proved that simply typable λ_{clos}-terms are strongly normalizing, but that any λ_{clos}-term that is mapped by h to a strongly normalizing λ-term, and all of whose subterms are also mapped by h to strongly normalizing λ-terms, is itself strongly normalizing (in λ_{clos}). Therefore, the result also holds for systems of conjunctive types without a universal type (system $\mathcal{D}$), of second-order types (system F), of positive recursive types, etc.

This also works to show that finite developments hold for λ_{clos}, i.e. that restricting (β)-reduction to only contract residuals of (β)-redexes already present in the initial term s_0 of the reduction (but leaving all other rules unconstrained) defines a terminating notion of reduction. Indeed, mark all closures $\langle x \cdot s, \theta\rangle$ in s_0 as $\langle x \cdot s, \theta\rangle^*$, and redefine (β) to only work when the closure subterm on the left-hand side is unmarked (marked closures can be thought as a second closure operator on which no (β) rule is defined, but to which (ι), (0) and (2) still apply). Then the same technique as above, using the fact that finite developments hold for the λ-calculus, shows that this new notion of reduction indeed terminates. Since this notion of reduction is also locally confluent (as a tedious check shows), it is also confluent. By standard techniques [Bar84], it follows that λ_{clos} is confluent.

Normal=Cut-free: We now show that, given any λ_{clos}-term s such that $\Gamma \vdash s : \tau$ is derivable in the natural deduction system of Figure 5, then there is a derivation of $\Gamma \vdash s : \tau$ in the sequent system of Figure 6 that does not use rule (Cut). This is by structural induction on s.

First, notice that: *(iv)* whenever $\Gamma, y : \tau \vdash s : \tau'$, $\Gamma \vdash s_1 : \tau_1$, $\ldots$, $\Gamma \vdash s_n$ all have cut-free derivations in the system of Figure 6, say π, π_1, $\ldots$, π_n respectively, and the binding $x : \tau_1 \to \ldots \to \tau_n \to \tau$ belongs to Γ, then $\Gamma \vdash s[xs_1 \dots s_n/y] : \tau'$ also has a cut-free derivation in the system of Figure 6. Indeed, this is clear if $n = 0$; if $n \geq 1$, just take:

$$
\dfrac{\begin{array}{c}\dfrac{\dfrac{\overset{\vdots\,\pi}{\Gamma, \Gamma_n \vdash s : \tau'} \quad \overset{\vdots\,\pi_n}{\Gamma, \Gamma_{n-1} \vdash s_n : \tau_n}}{\Gamma, \Gamma_{n-1} \vdash s[y_{n-1}s_n/y_n] : \tau'} \qquad \overset{\vdots\,\pi_{n-1}}{\Gamma, \Gamma_{n-2} \vdash s_{n-1} : \tau_{n-1}}}{\Gamma, \Gamma_{n-2} \vdash s[y_{n-2}s_{n-1}s_n/y_n] : \tau'} \\ \vdots \\ \Gamma, \Gamma_1 \vdash y_1 s_2 \dots s_n : \tau \end{array} \qquad \overset{\vdots\,\pi_1}{\Gamma, \Gamma_0 \vdash s_1 : \tau_1}}{\Gamma, \Gamma_0 \vdash s[y_0 s_1 \dots s_n/y_n] : \tau'}
$$

where we take y_0 to denote x, y_n to denote y, y_1, ..., y_{n-1} to be fresh, pairwise distinct variables, where we have used Γ_i as an abbreviation for the context $y_1 : \tau_2 \to \dots \to \tau_n \to \tau, y_2 : \tau_3 \to \dots \to \tau_n \to \tau, \dots, y_i : \tau_{i+1} \to \dots \to \tau_n \to \tau$, for every i, $0 \leq i \leq n$ (Γ_0 is empty), and where we have used weakening on π, π_2, ..., π_n silently.

If s is a variable x, then $x : \tau$ must be in Γ, so we obtain the following valid cut-free proof:

$$
\dfrac{}{\Gamma \vdash x : \tau}
$$

If s is an application, then it must be of the form $xs_1 \dots s_n$ for some variable x. Moreover, Γ must contain an assumption of the form $x : \tau_1 \to \dots \to \tau_n \to \tau$, and $\Gamma \vdash s_i : \tau_i$ is derivable in the natural deduction system, for all i, $1 \leq i \leq n$. By induction hypothesis, we have cut-free proofs π_i of $\Gamma \vdash s_i : \tau_i$, $1 \leq i \leq n$, and we use (iv) on the latter plus the following proof π:

$$
\dfrac{}{\Gamma, y : \tau \vdash y : \tau}
$$

Finally, if s is a closure, then s must be of the form $\langle x \cdot t, \{x_1 := t_1, \dots, x_k := t_k\}\rangle$, where: (i) $x_1, \dots, x_k$ are exactly the free variables of t other than x (otherwise rule (0) would apply); (ii) $t_1, \dots, t_k$ are normal typable terms that are not closures (otherwise rule (ι) would apply), so that for every i, $1 \leq i \leq k$, t_i is of the form $y_i t_{i1} \dots t_{in_i}$, with y_i a variable. Note that because of (i), $\langle x \cdot t\rangle$ is exactly $\langle x \cdot t, \{x_1 := x_1, \dots, x_k := x_k\}\rangle$, and therefore s is exactly $\langle x \cdot t\rangle[t_1/x_1, \dots, t_k/x_k]$, or equivalently, and because of the variable naming convention, $\langle x \cdot t\rangle[t_1/x_1][t_2/x_2] \dots [t_k/x_k]$. Now $\Gamma \vdash s : \tau$ was derivable in the natural deduction system, first τ must be of the form $\tau' \to \tau''$, and second we get as a subderivation $x_1 : \tau_1, \dots, x_k : \tau_k, x : \tau' \vdash t : \tau''$, so by induction hypothesis we get a cut-free proof π of the same sequent. On the other hand, by induction hypothesis again, there are cut-free proofs π_{ij} of $\Gamma \vdash t_{ij} : \tau_{ij}$, $1 \leq i \leq k$, $1 \leq j \leq n_i$, and $y_i : \tau_{i1} \to \dots \to \tau_{in_i} \to \tau_i$ is in Γ, $1 \leq i \leq k$. We then build:

$$
\dfrac{\begin{array}{c}\dfrac{\dfrac{\overset{\vdots\,\pi}{\Gamma, x_1 : \tau_1, \dots, x_k : \tau_k, x : \tau' \vdash t : \tau''}}{\Gamma, x_1 : \tau_1, \dots, x_k : \tau_k \vdash \langle x \cdot t\rangle : \tau} \quad \vdots\,\pi_{11} \quad \dots \quad \vdots\,\pi_{1n_1}}{\Gamma, x_2 : \tau_2, \dots, x_k : \tau_k \vdash \langle x \cdot t\rangle[t_1/x_1] : \tau}\text{ by }(iv) \\ \vdots \\ \Gamma, x_k : \tau_k \vdash \langle x \cdot t\rangle[t_1/x_1] \dots [t_{k-1}/x_{k-1}] : \tau \end{array} \quad \vdots\,\pi_{k1} \quad \dots \quad \vdots\,\pi_{kn_k}}{\Gamma \vdash \langle x \cdot t\rangle[t_1/x_1] \dots [t_{k-1}/x_{k-1}][t_k/x_k] : \tau}\text{ by }(iv)
$$

where again weakenings have been used implicitly.

Cut-elimination Let $\Gamma \vdash s : \tau$ be derivable in the sequent system of Figure 6, or equivalently in the natural deduction system of Figure 5. Then s is strongly normalizing: let t be one of its normal forms (actually, its unique normal form, since λ_{clos} is confluent), then by subject reduction $\Gamma \vdash t : \tau$ is derivable in the sequent system, and by the above, we don't need to use (Cut). Moreover $s \longrightarrow^* t$.

All this also works in the presence of the η-rule. In particular, strong normalization follows from the following theorem [Der87]: Let $\xrightarrow{R_1}$ and $\xrightarrow{R_2}$ be two rewrite relations. $\xrightarrow{R_1}$ is said to *quasi-commute* over $\xrightarrow{R_2}$ if and only if for every pair of rewrite steps of the form:

$$u \xrightarrow{R_1} v \xrightarrow{R_2} w$$

there is another sequence of rewrite steps from u to w of the form:

$$u \xrightarrow{R_2} v' \xrightarrow{R_1 \cup R_2}{}^{*} w$$

where $\xrightarrow{R_1 \cup R_2}{}^{*}$ denotes any finite number of $\xrightarrow{R_1}$ and $\xrightarrow{R_2}$ steps. Then $\xrightarrow{R_1 \cup R_2}$ terminates if and only if both $\xrightarrow{R_1}$ and $\xrightarrow{R_2}$ terminate. Then, the simply-typed $\lambda_{\mathrm{clos}\,\eta}$-calculus terminates because its η-rule quasi-commutes over (β), (ι), (0) and (2), and η on its own clearly terminates. (Notice that η is right-linear, and all other rules are left-linear, so that we need only consider superpositions between the right-hand side of η and the left-hand sides of the other rules; this is an easy check.)

The $\lambda_{\mathrm{clos}\,\eta}$ is also confluent, by the standard Hindley-Rosen argument [Bar84]: η along is confluent, because it is orthogonal (left-linear without critical pairs), and η commutes in the following strong sense: if s rewrites in one η step to t, or in one step of some other rule R to t', then an easy check shows that t contracts in one R step (contracting the residual of the R-redex leading to t' in s if it is still present) or is equal to some term t'' (if the residual disappeared from s), and that t' reduces in zero, one or more η steps to t'' (contracting all residuals of the η-redex of s in t').

Notice then that the effect of the η-rule on sequent proofs is to transform the cut-free derivations of the form:

$$\dfrac{\dfrac{\dfrac{}{\Gamma, y : \tau_2, x : \tau_1 \vdash y : \tau_2} \quad \dfrac{}{\Gamma, x : \tau_1 \vdash x : \tau_1}}{\Gamma, z : \tau_1 \to \tau_2, x : \tau_1 \vdash zx : \tau_2}}{\Gamma, z : \tau_1 \to \tau_2, \Delta \vdash \langle x \cdot zx \rangle : \tau_1 \to \tau_2}$$

into the simpler:

$$\dfrac{}{\Gamma, z : \tau_1 \to \tau_2, \Delta \vdash z : \tau_1 \to \tau_2}$$

D Proof of Theorem 3

Item (i): if $\Gamma \vdash s : \tau$ is derivable in the simple type discipline for λ_{clos}, then we show by structural induction on s that $\Gamma \vdash s^* : \tau$ in the simple type discipline for SKInT. All cases except when s is a closure are trivial, so assume that $s = \langle x \cdot t, \{x_1 := t_1, \ldots, x_k := t_k\}\rangle$, where $x_1 : \tau'_1 \to \tau''_1, \ldots, x_k : \tau'_k \to \tau''_k, x : \tau' \vdash t : \tau''$, with $\tau = \tau' \to \tau''$, and $\Gamma \vdash t_i : \tau'_i \to \tau''_i$ are derivable, $1 \le i \le k$. By induction hypothesis, the same sequents with t^* and t^*_i instead of t and t_i respectively are derivable in SKInT. But an easy induction on t^* then shows that $x_1 : \tau'_1 \to \tau''_1, \ldots, x_k : \tau'_k \to \tau''_k \vdash [x]t^* : \tau' \to \tau''$ is derivable in SKInT. Since (Cut) is admissible in the simple type discipline for SKInT, it follows that $\Gamma \vdash [x]t^*[t^*_1/x_1, \ldots, t^*_k/x_k] : \tau' \to \tau''$ is also derivable. This concludes the proof.

Item (ii): if $\Gamma \vdash u : \tau$ is derivable in the simple type discipline for SKInT, then again we show by structural induction on u that $\Gamma \vdash u^\circ : \tau$ is derivable in the simple type discipline for λ_{clos}. This is obvious if u is a variable. If u is I_ℓ of type $\tau_0 \to \ldots \to \tau_{\ell-1} \to \tau \to \tau$, then we produce the following derivation for u°:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{z : \tau \vdash z : \tau}}{x_{\ell-1} : \tau_{\ell-1} \vdash \langle z \cdot z \rangle : \tau \to \tau}}{x_{\ell-2} : \tau_{\ell-2} \vdash \langle x_{\ell-1} \cdot \langle z \cdot z \rangle\rangle : \tau_{\ell-1} \to \tau \to \tau}}{\vdots}}{\dfrac{x_1 : \tau_1 \vdash \langle x_2 \cdot \ldots \langle x_{\ell-1} \cdot \langle z \cdot z \rangle\rangle \ldots \rangle : \tau_2 \to \ldots \to \tau_{\ell-1} \to \tau \to \tau}{\Gamma \vdash \langle x_1 \cdot \ldots \langle x_{\ell-1} \cdot \langle z \cdot z \rangle\rangle \ldots \rangle : \tau_1 \to \ldots \to \tau_{\ell-1} \to \tau \to \tau}}$$

If u is of the form $S_\ell(u_1, u_2)$, then τ is of the form $\tau_0 \to \ldots \to \tau_{\ell-1} \to \tau'$, where u_1 is of type $\tau_0 \to \ldots \to \tau_{\ell-1} \to \tau_\ell \to \tau'$ for some τ_ℓ, and u_2 is of type $\tau_0 \to \ldots \to \tau_{\ell-1} \to \tau_\ell$. So it suffices to show that:

$$\frac{\Gamma \vdash s : \tau_0 \to \ldots \to \tau_{\ell-1} \to \tau_\ell \to \tau' \quad \Gamma \vdash t : \tau_0 \to \ldots \to \tau_{\ell-1} \to \tau_\ell}{\Gamma \vdash \mathcal{S}_\ell(s,t) : \tau_0 \to \ldots \to \tau_{\ell-1} \to \tau'}$$

is a derived rule in the simple type system for λ_{clos}. This is by induction on ℓ; the result is clear when $\ell = 0$, otherwise, letting Δ be the context $x : \tau_0 \to \ldots \to \tau_{\ell-1} \to \tau_\ell \to \tau', y : \tau_0 \to \ldots \to \tau_{\ell-1} \to \tau_\ell$, we have:

$$\dfrac{\dfrac{\dfrac{}{\begin{array}{c}\Delta, z : \tau_0 \vdash x : \tau_0 \to \ldots \\ \to \tau_{\ell-1} \to \tau_\ell \to \tau'\end{array}} \quad \dfrac{}{\Delta, z : \tau_0 \vdash z : \tau_0}}{\Delta, z : \tau_0 \vdash xz : \tau_1 \to \ldots \to \tau_{\ell-1} \to \tau_\ell \to \tau'} \quad \dfrac{\dfrac{}{\begin{array}{c}\Delta, z : \tau_0 \vdash y : \tau_0 \to \ldots \\ \to \tau_{\ell-1} \to \tau_\ell\end{array}} \quad \dfrac{}{\Delta, z : \tau_0 \vdash z : \tau_0}}{\Delta, z : \tau_0 \vdash yz : \tau_1 \to \ldots \to \tau_{\ell-1} \to \tau_\ell}}{\dfrac{\Delta, z : \tau_0 \vdash \mathcal{S}_{\ell-1}(xz, yz) : \tau_1 \to \ldots \to \tau_{\ell-1} \to \tau'}{\Gamma \vdash \langle z \cdot \mathcal{S}_{\ell-1}(xz, yz), \{x := s, y := t\}\rangle : \tau_0 \to \ldots \to \tau_{\ell-1} \to \tau'}} \text{(by induction)}$$

where the last line also uses the given derivations of $\Gamma \vdash s : \tau_0 \to \ldots \to \tau_{\ell-1} \to \tau_\ell \to \tau'$ and $\Gamma \vdash t : \tau_0 \to \ldots \to \tau_{\ell-1} \to \tau_\ell$. Similarly, for $K_\ell(u)$, we have to show that:

$$\frac{\Gamma \vdash s : \tau_0 \to \ldots \to \tau_{\ell-1} \to \tau' \to \tau''}{\Gamma \vdash \mathcal{K}_\ell(s) : \tau_0 \to \ldots \to \tau_{\ell-1} \to \tau \to \tau' \to \tau''}$$

is a derived rule of the simple type system for λ_{clos}. When $\ell = 0$, we have:

$$\frac{\Gamma \vdash s : \tau' \to \tau'' \quad \overline{x : \tau' \to \tau'', z : \tau \vdash x : \tau' \to \tau''}}{\Gamma \vdash \langle z \cdot x, \{x := s\}\rangle : \tau \to \tau' \to \tau''}$$

which works just because $\tau' \to \tau''$ is an arrow type. Then case $\ell \geq 1$ works mostly as the similar case for $\mathcal{S}_\ell$.

Item (*iii*): Recall that [GGL98]: (a) $S_0([x]u, v) \longrightarrow^+ u[v/x]$ in SKInT, (b) if u rewrites to v in SKInT (resp. SKInT$_\eta$), then $[x]u$ rewrites to $[x]v$. By (b), in particular, it suffices to check each rule in turn, then the result will follow.

(β) $(\langle x \cdot t, \{x_1 := t_1, \ldots, x_k := t_k\}\rangle s) = S_0([x]t^*[t_1^*/x_1, \ldots, t_k^*/x_k], s^*) \longrightarrow^+ t^*[t_1^*/x_1, \ldots, t_k^*/x_k][s^*/x]$ (by (a)) $= t^*[t_1^*/x_1, \ldots, t_k^*/x_k, s^*/x]$ (by the variable naming condition, and the easy fact that e^* has the same free variables as e for all e). Now we claim that the latter equals $(t[t_1/x_1, \ldots, t_k/x_k, s/x])^*$.

In general, we show that: (c) for any t, t_1, …, t_k, $t^*[t_1^*/x_1, \ldots, t_k^*/x_k] = (t[t_1/x_1, \ldots, t_k/x_k])^*$, by structural induction on t; the cases when t is a variable or an application are clear, so let t be $\langle x' \cdot t', \{y_1 := t'_1, \ldots, y_p := t'_p\}\rangle$: then:

$$\begin{array}{l}
t^*[t_1^*/x_1, \ldots, t_k^*/x_k] \\
= ([x']t'^*)[{t'_1}^*/y_1, \ldots, {t'_p}^*/y_p][t_1^*/x_1, \ldots, t_k^*/x_k] \\
= ([x']t'^*)[{t'_1}^*/y_1, \ldots, {t'_p}^*/y_p][t_1^*/x_1, \ldots, t_k^*/x_k] \\
= ([x']t'^*)[{t'_1}^*[t_1^*/x_1, \ldots, t_k^*/x_k]/y_1, \ldots, {t'_p}^*[t_1^*/x_1, \ldots, t_k^*/x_k]/y_p] \\
\quad \text{(because all the free variables of } t'\text{, hence of } t'^*\text{,} \\
\quad \text{are among } y_1, \ldots, y_p \text{ by definition of closures)} \\
= ([x']t'^*)[(t'_1[t_1/x_1, \ldots, t_k/x_k])^*/y_1, \ldots, (t'_p[t_1/x_1, \ldots, t_k/x_k])^*/y_p] \\
\quad \text{(by induction)} \\
= \langle x' \cdot t', \{y_1 := t'_1[t_1/x_1, \ldots, t_k/x_k]/y_1, \ldots, y_p := t'_p[t_1/x_1, \ldots, t_k/x_k]\}\rangle^* \\
= (t[t_1/x_1, \ldots, t_k/x_k])^*
\end{array}$$

(ι) We have:

$$\begin{array}{l}
\langle x \cdot t, \{x_1 := \langle y \cdot s, \{y_1 := s_1, \ldots, y_p := s_p\}\rangle, x_2 := t_2, \ldots, x_k := t_k\}\rangle^* \\
= ([x]t^*)[([y]s^*)[s_1^*/y_1, \ldots, s_p^*/y_p]/x_1, t_2^*/x_2, \ldots, t_k^*/x_k] \\
= ([x]t^*)[[y]s^*/x_1][s_1^*/y_1, \ldots, s_p^*/y_p, t_2^*/x_2, \ldots, t_k^*/x_k]
\end{array}$$

Now call a SKInT-term a *value* if and only if it does not contain any term of the form $S_0(u_1, u_2)$. Observe that: (d) for every value V, for every variable y not free in V, $K_0(V) \longrightarrow^* [y]V$ in SKInT(straightforward structural induction on V).

It follows that: (e) for every distinct variables x, y, for every SKInT-term u and every SKInT-value V, if y is not free in V, then $([y]u)[V/x] \longrightarrow^* [y](u[V/x])$. This is by structural induction on u: if $u = x$, this is by (d); if $u = y$, then both sides are I_0; if u is any other variable, then both sides are $K_0(u)$; in all other cases, this is by a straightforward appeal to the induction hypothesis.

Also, observe that: (f) for every SKInT-term v, $[y]v$ is a value. This is trivial.

We conclude that $([]t^*)[[y]s^*/x_1] \longrightarrow^* [x](t^*[[y]s^*/x_1])$ (by (f) and (e)) $= [x](t^*[\langle y \cdot s\rangle^*/x_1]) = [x](t[\langle y \cdot s\rangle/x_1])^*$ (by (c)). It follows that:

$$\begin{aligned}
&\langle x \cdot t, \{x_1 := \langle y \cdot s, \{y_1 := s_1, \ldots, y_p := s_p\}\rangle, x_2 := t_2, \ldots, x_k := t_k\}\rangle^* \\
&\longrightarrow^* ([x](t[\langle y \cdot s\rangle/x_1])^*)[s_1^*/y_1, \ldots, s_p^*/y_p, t_2^*/x_2, \ldots, t_k^*/x_k] \\
&= (\langle x \cdot t[\langle y \cdot s\rangle/x_1], \{y_1 := s_1, \ldots, y_p := s_p, x_2 := t_2, \ldots, x_k := t_k\}\rangle)^*
\end{aligned}$$

(0) If x_1 is not free in t, then $\langle x \cdot t, \{x_1 := t_1, x_2 := t_2, \ldots, x_k := t_k\}\rangle^* = ([x]t^*)[t_1^*/x_1, t_2^*/x_2, \ldots, t_k^*/x_k] = ([x]t^*)[t_2^*/x_2, \ldots, t_k^*/x_k]$ (because x_1 cannot be free in t^* either, hence in $[x]t^*$) $= \langle x \cdot t, \{x_2 := t_2, \ldots, x_k := t_k\}\rangle^*$.

(2) If $t_1 = t_2$ (up to α-renaming), then it is easy to see that $t_1^* = t_2^*$ (textually): first show by structural induction on u that $[x]u = [y](u[y/x])$, where y is not free in u, then the result follows by structural induction on t_1. So $\langle x \cdot t, \{x_1 := t_1, x_2 := t_2, \ldots, x_k := t_k\}\rangle^* = ([x]t^*)[t_1^*/x_1, t_2^*/x_2, \ldots, t_k^*/x_k] = ([x]t^*)[x_2/x_1][t_2^*/x_2, \ldots, t_k^*/x_k]$. But an easy structural induction on u shows that $([x]u)[x_2/x_1] = [x](u[x_2/x_1])$ as soon as $x \neq x_1$ and $x \neq x_2$, and another structural induction on t then shows that $t^*[x_2/x_1] = (t[x_2/x_1])^*$. So $\langle x \cdot t, \{x_1 := t_1, x_2 := t_2, \ldots, x_k := t_k\}\rangle^* = ([x](t[x_2/x_1])^*)[t_2^*/x_2, \ldots, t_k^*/x_k] = \langle x \cdot t[x_2/x_1], \{x_2 := t_2, \ldots, x_k := t_k\}\rangle^*$.

(η) In the case of η-reduction, $\langle x \cdot zx, \theta\rangle^* = ([x]S_0(z, x))[(z\hat{\theta})^*/z] = S_1(K_0((z\hat{\theta})^*), I_0)$ rewrites by one application of the SKInT η-rule to $(z\hat{\theta})^*$.

Item (*iv*): Observe first that we can define $I_\ell{}^\circ$ by induction on ℓ as:

$$\begin{aligned}
I_0{}^\circ &=_{\mathrm{df}} \langle z \cdot z\rangle \\
I_{\ell+1}{}^\circ &=_{\mathrm{df}} \langle z \cdot I_\ell{}^\circ\rangle
\end{aligned}$$

As the $u \mapsto u^\circ$ translation clearly passes to context, it is enough to consider each rule in turn:

- Rule (SI_ℓ): $(S_\ell(I_\ell, w))^\circ = \mathcal{S}_\ell(I_\ell{}^\circ, w^\circ)$. We claim that this rewrites to w°; to this end, we show by induction on ℓ that $\mathcal{S}_\ell(I_\ell{}^\circ, t) \longrightarrow_\eta^* t$ for all λ_{clos}-terms t.

  If $\ell = 0$, then $\mathcal{S}_0(I_0{}^\circ, t) = \langle z \cdot z\rangle t \longrightarrow t$ by (β). Induction case:

$$\begin{aligned}
&\mathcal{S}_{\ell+1}(I_{\ell+1}{}^\circ, t) \\
&= \langle z \cdot \mathcal{S}_\ell(xz, yz), \{x := \langle z' \cdot I_\ell{}^\circ\rangle, y := t\}\rangle \\
&\longrightarrow \langle z \cdot \mathcal{S}_\ell(\langle z' \cdot I_\ell{}^\circ\rangle z, yz), \{y := t\}\rangle && \text{(by } (\iota)\text{)} \\
&\longrightarrow \langle z \cdot \mathcal{S}_\ell(I_\ell{}^\circ, yz), \{y := t\}\rangle && \text{(by } (\beta)\text{)} \\
&\approx \langle z \cdot yz, \{y := t\}\rangle && \text{(by induction)} \\
&\longrightarrow_\eta t && \text{(by } (\eta)\text{)}
\end{aligned}$$

- Rule (SK_ℓ): $(S_\ell(K_\ell(u), w))^\circ = \mathcal{S}_\ell(\mathcal{K}_\ell(u^\circ), w^\circ)$. We check that, in general, $\mathcal{S}_\ell(\mathcal{K}_\ell(s), t) \longrightarrow_\eta^* s$, by induction on ℓ.

If $\ell = 0$, then $\mathcal{S}_0(\mathcal{K}_0(s), t) = \langle z \cdot x, \{x := s\}\rangle t \longrightarrow s$. Otherwise:

$$\begin{array}{ll}
\mathcal{S}_{\ell+1}(\mathcal{K}_{\ell+1}(s), t) & \\
= \langle z \cdot \mathcal{S}_\ell(xz, yz), \{x := \langle z' \cdot \mathcal{K}_\ell(x', z'), \{x' := s\}\rangle, y := t\}\rangle & \\
\longrightarrow \langle z \cdot \mathcal{S}_\ell(\langle z' \cdot \mathcal{K}_\ell(x'z')\rangle z, yz), \{x' := s, y := t\}\rangle & \text{(by } (\iota)) \\
\longrightarrow \langle z \cdot \mathcal{S}_\ell(\mathcal{K}_\ell(x'z), yz), \{x' := s, y := t\}\rangle & \text{(by } (\beta)) \\
\longrightarrow_\eta^* \langle z \cdot x'z, \{x' := s, y := t\}\rangle & \text{(by induction)} \\
\longrightarrow_\eta s & \text{(by } (\eta))
\end{array}$$

- Rule $(S_\ell I_\mathcal{L})$: it suffices to show that $\mathcal{S}_\ell(I_\mathcal{L}{}^\circ, t) \longrightarrow^* I_{\mathcal{L}-1}{}^\circ$ for any λ_{clos}-term t. If $\ell = 0$, then the left-hand side is $\langle z \cdot I_{\mathcal{L}-1}{}^\circ\rangle t \longrightarrow I_{\mathcal{L}-1}^\circ$ by (β). Otherwise:

$$\begin{array}{ll}
\mathcal{S}_{\ell+1}(I_{\mathcal{L}+1}{}^\circ, t) & \\
= \langle z \cdot \mathcal{S}_\ell(xz, yz), \{x := \langle z' \cdot I_\mathcal{L}{}^\circ\rangle, y := t\}\rangle & \\
\longrightarrow \langle z \cdot \mathcal{S}_\ell(\langle z' \cdot I_\mathcal{L}{}^\circ\rangle z, yz), \{y := t\}\rangle & \text{(by } (\iota)) \\
\longrightarrow \langle z \cdot \mathcal{S}_\ell(I_\mathcal{L}{}^\circ, yz), \{y := t\}\rangle & \text{(by } (\beta)) \\
\longrightarrow^* \langle z \cdot I_{\mathcal{L}-1}{}^\circ, \{y := t\}\rangle & \text{(by induction)} \\
\longrightarrow \langle z \cdot I_{\mathcal{L}-1}{}^\circ\rangle & \text{(by (0))} \\
= I_\mathcal{L}{}^\circ &
\end{array}$$

- Rule $(S_\ell S_\mathcal{L})$; it suffices to show that $\mathcal{S}_\ell(\mathcal{S}_\mathcal{L}(r, s), t) \approx \mathcal{S}_{\mathcal{L}-1}(\mathcal{S}_\ell(r, t), \mathcal{S}_\ell(s, t))$ for all λ_{clos}-terms r, s, t. If $\ell = 0$, then the left-hand side is $\langle z \cdot \mathcal{S}_{\mathcal{L}-1}(xz, yz), \{x := r, y := s\}\rangle t \longrightarrow \mathcal{S}_{\mathcal{L}-1}(rt, st)$ (by (β)), and the latter equals the right-hand side. Otherwise:

$$\begin{array}{ll}
\mathcal{S}_{\ell+1}(\mathcal{S}_{\mathcal{L}+1}(r, s), t) & \\
= \langle z \cdot \mathcal{S}_\ell(xz, yz), \{x := \langle z' \cdot \mathcal{S}_\mathcal{L}(x'z', y'z'), \{x' := r, y' := s\}\rangle, y := t\}\rangle & \\
\longrightarrow \langle z \cdot \mathcal{S}_\ell(\langle z' \cdot \mathcal{S}_\mathcal{L}(x'z', y'z'), \{x' := x', y' := y'\}\rangle z, yz), \{x' := r, y' := s, y := t\}\rangle & \text{(by } (\iota)) \\
\longrightarrow \langle z \cdot \mathcal{S}_\ell(\mathcal{S}_\mathcal{L}(x'z, y'z), yz), \{x' := r, y' := s, y := t\}\rangle & \text{(by } (\beta))
\end{array}$$

while:

$$\begin{array}{ll}
\mathcal{S}_\mathcal{L}(\mathcal{S}_{\ell+1}(r, t), \mathcal{S}_{\ell+1}(s, t)) & \\
= \langle z \cdot \mathcal{S}_{\mathcal{L}-1}(xz, yz)\{x := \langle z' \cdot \mathcal{S}_\ell(x'z', y'z'), \{x' := r, y' := t\}\rangle, & \\
\qquad\qquad y := \langle z'' \cdot \mathcal{S}_\ell(x''z'', y''z''), \{x'' := s, y'' := t\}\rangle\} & \\
\longrightarrow^+ \langle z \cdot \mathcal{S}_{\mathcal{L}-1}(\langle z' \cdot \mathcal{S}_\ell(x'z', y'z'), \{x' := x', y' := y'\}\rangle z, & \\
\qquad\qquad \langle z'' \cdot \mathcal{S}_\ell(x''z'', y''z''), \{x'' := x'', y'' := y''\}\rangle z) & \\
\qquad \{x' := r, y' := t, x'' := s, y'' := t\}\rangle & \text{(by } (\iota)) \\
\longrightarrow^+ \langle z \cdot \mathcal{S}_{\mathcal{L}-1}(\mathcal{S}_\ell(x'z, y'z), \mathcal{S}_\ell(x''z, y''z)), \{x' := r, y' := t, x'' := s, y'' := t\}\rangle & \text{(by } (\beta)) \\
\longrightarrow \langle z \cdot \mathcal{S}_{\mathcal{L}-1}(\mathcal{S}_\ell(x'z, y'z), \mathcal{S}_\ell(x''z, y'z)), \{x' := r, y' := t, x'' := s\}\rangle & \text{(by (2))} \\
= \langle z \cdot \mathcal{S}_{\mathcal{L}-1}(\mathcal{S}_\ell(x'z, yz), \mathcal{S}_\ell(y'z, yz)), \{x' := r, y' := s, y := t\}\rangle & \text{(by } \alpha\text{-renaming)}
\end{array}$$

But now the end terms of each reduction sequence are convertible by $\approx$ by induction hypothesis.

- Rule $(S_\ell K_\mathcal{L})$; it suffices to show that $\mathcal{S}_\ell(\mathcal{K}_\mathcal{L}(s), t) \approx \mathcal{K}_{\mathcal{L}-1}(\mathcal{S}_\ell(s, t))$ for all λ_{clos}-terms s, t. If $\ell = 0$, then the left-hand side is $\langle z \cdot \mathcal{K}_{\mathcal{L}-1}(xz), \{x := s\}\rangle t \longrightarrow \mathcal{K}_{\mathcal{L}-1}(st)$ (by (β)), and the latter equals the right-hand side. Otherwise:

$$\begin{array}{ll}
\mathcal{S}_{\ell+1}(\mathcal{K}_{\mathcal{L}+1}(s), t) & \\
= \langle z \cdot \mathcal{S}_\ell(xz, yz), \{x := \langle z' \cdot \mathcal{K}_\mathcal{L}(x'z'), \{x' := s\}\rangle, y := t\}\rangle & \\
\longrightarrow \langle z \cdot \mathcal{S}_\ell(\langle z' \cdot \mathcal{K}_\mathcal{L}(x'z'), \{x' := x'\}\rangle z, yz), \{x' = s, y := t\}\rangle & \text{(by } (\iota)) \\
\longrightarrow \langle z \cdot \mathcal{S}_\ell(\mathcal{K}_\mathcal{L}(x'z), yz), \{x' = s, y := t\}\rangle & \text{(by } (\iota))
\end{array}$$

while:

$$\begin{array}{ll}
\mathcal{K}_\mathcal{L}(\mathcal{S}_{\ell+1}(s, t)) & \\
= \langle z \cdot \mathcal{K}_{\mathcal{L}-1}(xz), \{x := \langle z' \cdot \mathcal{S}_\ell(x'z', y'z'), \{\rangle x' := s, y' := t\}\rangle & \\
\longrightarrow \langle z \cdot \mathcal{K}_{\mathcal{L}-1}(\langle z' \cdot \mathcal{S}_\ell(x'z', y'z'), \{x' := x', y' := y'\}\rangle z), \{x' := s, y' := t\}\rangle & \text{(by } (\iota)) \\
\longrightarrow \langle z \cdot \mathcal{K}_{\mathcal{L}-1}(\mathcal{S}_\ell(x'z, y'z)), \{x' := s, y' := t\}\rangle & \text{(by } (\iota)) \\
\longrightarrow \langle z \cdot \mathcal{K}_{\mathcal{L}-1}(\mathcal{S}_\ell(x'z, yz)), \{x' := s, y := t\}\rangle & \text{(by } \alpha\text{-renaming)}
\end{array}$$

And the end terms of each reduction sequence are convertible by $\approx$ by induction hypothesis.

- Rule $(K_\ell I_\mathcal{L})$: we show that $\mathcal{K}_\ell(I_{\mathcal{L}-1}{}^\circ) \longrightarrow^* I_\mathcal{L}{}^\circ$. If $\ell = 0$, then the left-hand side is $\langle z \cdot x, \{x := I_{\mathcal{L}-1}{}^\circ\}\rangle \longrightarrow \langle z \cdot I_{\mathcal{L}-1}{}^\circ\rangle$ (by (ι), noticing that $I_{\mathcal{L}-1}{}^\circ$ is always a closure) $= I_\mathcal{L}{}^\circ$. Otherwise:

$$\begin{array}{ll}
\mathcal{K}_{\ell+1}(I_\mathcal{L}{}^\circ) & \\
= \langle z \cdot \mathcal{K}_\ell(xz), \{x := \langle z' \cdot I_{\mathcal{L}-1}{}^\circ\rangle\}\rangle & \\
\longrightarrow \langle z \cdot \mathcal{K}_\ell(\langle z' \cdot I_{\mathcal{L}-1}{}^\circ\rangle z)\rangle & \text{(by } (\iota))\\
\longrightarrow \langle z \cdot \mathcal{K}_\ell(I_{\mathcal{L}-1}{}^\circ)\rangle & \text{(by } (\beta))\\
\longrightarrow^* \langle z \cdot I_\mathcal{L}{}^\circ\rangle & \text{(by induction hypothesis)}\\
= I_{\mathcal{L}+\infty}{}^\circ &
\end{array}$$

- Rule $(K_\ell S_{\mathcal{L}+1})$: we show that $\mathcal{K}_\ell(\mathcal{S}_\mathcal{L}(s,t)) \approx \mathcal{S}_{\mathcal{L}+1}(\mathcal{K}_\ell(s), \mathcal{K}_\ell(t))$. When $\ell = 0$:

$$\begin{array}{ll}
\mathcal{K}_0(\mathcal{S}_\mathcal{L}(s,t)) & \\
= \langle z \cdot x, \{x := \langle z' \cdot \mathcal{S}_{\mathcal{L}-1}(x'z', y'z'), \{x' := s, y' := t\}\rangle\}\rangle & \\
\longrightarrow \langle z \cdot \langle z' \cdot \mathcal{S}_{\mathcal{L}-1}(x'z', y'z'), \{x' := x', y' := y'\}\rangle, \{x' := s, y' := t\}\rangle & \text{(by } (\iota))\\
= \langle z \cdot \mathcal{S}_\mathcal{L}(x', y'), \{x' := s, y' := t\}\rangle &
\end{array}$$

while:

$$\begin{array}{ll}
\mathcal{S}_{\mathcal{L}+1}(\mathcal{K}_0(s), \mathcal{K}_0(t)) & \\
= \langle z \cdot \mathcal{S}_\mathcal{L}(xz, yz), \{x := \langle z' \cdot x', \{x' := s\}\rangle, y := \langle z'' \cdot y', \{y' := t\}\rangle\}\rangle & \\
\longrightarrow^+ \langle z \cdot \mathcal{S}_\mathcal{L}(\langle z' \cdot x', \{x' := x'\}\rangle z, \langle z'' \cdot y', \{y' := y'\}\rangle z), \{x' := s, y' := t\}\rangle & \text{(by } (\iota))\\
\longrightarrow^+ \langle z \cdot \mathcal{S}_\mathcal{L}(x', y'), \{x' := s, y' := t\}\rangle & \text{(by } (\beta))
\end{array}$$

Otherwise:

$$\begin{array}{ll}
\mathcal{K}_{\ell+1}(\mathcal{S}_{\mathcal{L}+1}(s,t)) & \\
= \langle z \cdot \mathcal{K}_\ell(xz), \{x := \langle z' \cdot \mathcal{S}_\mathcal{L}(x'z', y'z'), \{x' := s, y' := t\}\rangle\}\rangle & \\
\longrightarrow \langle z \cdot \mathcal{K}_\ell(\langle z' \cdot \mathcal{S}_\mathcal{L}(x'z', y'z'), \{x' := x', y' := y'\}\rangle z), \{x' := s, y' := t\}\rangle & \text{(by } (\iota))\\
\longrightarrow \langle z \cdot \mathcal{K}_\ell(\mathcal{S}_\mathcal{L}(x'z, y'z)), \{x' := s, y' := t\}\rangle & \text{(by } (\beta))
\end{array}$$

while:

$$\begin{array}{ll}
\mathcal{S}_{\mathcal{L}+2}(\mathcal{K}_{\ell+1}(s), \mathcal{K}_{\ell+1}(t)) & \\
= \langle z \cdot \mathcal{S}_{\mathcal{L}+1}(xz, yz), \{x := \langle z' \cdot \mathcal{K}_\ell(x'z'), \{x' := s\}\rangle, y := \langle z'' \cdot \mathcal{K}_\ell(y'z'), \{y' := t\}\rangle\}\rangle & \\
\longrightarrow^+ \langle z \cdot \mathcal{S}_{\mathcal{L}+1}(\langle z' \cdot \mathcal{K}_\ell(x'z'), \{x' := x'\}\rangle z, \langle z'' \cdot \mathcal{K}_\ell(y'z'), \{y' := y'\}\rangle z), \{x' := s, y' := t\}\rangle & \text{(by } (\iota))\\
\longrightarrow^+ \langle z \cdot \mathcal{S}_{\mathcal{L}+1}(\mathcal{K}_\ell(x'z), \mathcal{K}_\ell(y'z)), \{x' := s, y' := t\}\rangle & \text{(by } (\beta))
\end{array}$$

and both end terms are convertible by $\approx$ by induction hypothesis.

- Rule $(K_\ell K_\mathcal{L})$: we show that $\mathcal{K}_\ell(\mathcal{K}_{\mathcal{L}-1}(s)) \approx \mathcal{K}_\mathcal{L}(\mathcal{K}_\ell(s))$. If $\ell = 0$ and $\mathcal{L} = 1$, then:

$$\begin{array}{ll}
\mathcal{K}_0(\mathcal{K}_0(s)) & \\
= \langle z \cdot x, \{x := \langle z' \cdot x', \{x' := s\}\rangle\}\rangle & \\
\longrightarrow \langle z \cdot \langle z' \cdot x', \{x' := x'\}\rangle, \{x' := s\}\rangle & \text{(by } (\iota))\\
= \langle z \cdot \mathcal{K}_0(x'), \{x' := s\}\rangle &
\end{array}$$

while:

$$\begin{array}{ll}
\mathcal{K}_1(\mathcal{K}_0(s)) & \\
= \langle z \cdot \mathcal{K}_0(xz), \{x := \langle z' \cdot x', \{x' := s\}\rangle\}\rangle & \\
\longrightarrow \langle z \cdot \mathcal{K}_0(\langle z' \cdot x', \{x' := x'\}\rangle z), \{x' := s\}\rangle & \text{(by } (\iota))\\
\longrightarrow \langle z \cdot \mathcal{K}_0(x'), \{x' := s\}\rangle & \text{(by } (\beta))
\end{array}$$

If $\ell = 0$ and $\mathcal{L} \geq 2$, then:

$$\begin{array}{ll}
\mathcal{K}_0(\mathcal{K}_{\mathcal{L}-1}(s)) & \\
= \langle z \cdot x, \{x := \langle z' \cdot \mathcal{K}_{\mathcal{L}-2}(x'z'), \{x' := s\}\rangle\rangle & \\
\longrightarrow \langle z \cdot \langle z' \cdot \mathcal{K}_{\mathcal{L}-2}(x'z'), \{x' := x'\}\rangle, \{x' := s\}\rangle & \text{(by } (\iota))\\
= \langle z \cdot \mathcal{K}_{\mathcal{L}-1}(x'), \{x' := s\}\rangle &
\end{array}$$

while:

$$\begin{array}{ll}
\mathcal{K}_{\mathcal{L}}(\mathcal{K}_0(s)) & \\
= \langle z \cdot \mathcal{K}_{\mathcal{L}-1}(xz), \{x := \langle z' \cdot x', \{x' := s\}\rangle\}\rangle & \\
\longrightarrow \langle z \cdot \mathcal{K}_{\mathcal{L}-1}(\langle z' \cdot x', \{x' := x'\}\rangle z), \{x' := s\}\rangle & \text{(by } (\iota)) \\
\longrightarrow \langle z \cdot \mathcal{K}_{\mathcal{L}-1}(x'), \{x' := s\}\rangle & \text{(by } (\beta))
\end{array}$$

And if $\ell \geq 1$, then:

$$\begin{array}{ll}
\mathcal{K}_\ell(\mathcal{K}_{\mathcal{L}-1}(s)) & \\
= \langle z \cdot \mathcal{K}_{\ell-1}(xz), \{x := \langle z' \cdot \mathcal{K}_{\mathcal{L}-2}(x'z'), \{x' := s\}\rangle\}\rangle & \\
\longrightarrow \langle z \cdot \mathcal{K}_{\ell-1}(\langle z' \cdot \mathcal{K}_{\mathcal{L}-2}(x'z'), \{x' := x'\}\rangle z), \{x' := s\}\rangle & \text{(by } (\iota)) \\
\longrightarrow \langle z \cdot \mathcal{K}_{\ell-1}(\mathcal{K}_{\mathcal{L}-2}(x'z)), \{x' := s\}\rangle & \text{(by } (\beta))
\end{array}$$

while:

$$\begin{array}{ll}
\mathcal{K}_{\mathcal{L}}(\mathcal{K}_\ell(s)) & \\
= \langle z \cdot \mathcal{K}_{\mathcal{L}-1}(xz), \{x := \langle z' \cdot \mathcal{K}_{\ell-1}(x'z'), \{x' := s\}\rangle\}\rangle & \\
\longrightarrow \langle z \cdot \mathcal{K}_{\mathcal{L}-1}(\langle z' \cdot \mathcal{K}_{\ell-1}(x'z'), \{x' := x'\}\rangle z), \{x' := s\}\rangle & \text{(by } (\iota)) \\
\longrightarrow \langle z \cdot \mathcal{K}_{\mathcal{L}-1}(\mathcal{K}_{\ell-1}(x'z)), \{x' := s\}\rangle & \text{(by } (\beta))
\end{array}$$

and both terms are convertible by $\approx$, by induction hypothesis.

- Rule η: we show that $\mathcal{S}_{\ell+1}(\mathcal{K}_\ell(s), I_\ell{}^\circ) \longrightarrow^*_\eta s$ for any λ_{clos}-term s. When $\ell = 0$, $\mathcal{S}_1(\mathcal{K}_0(s), I_0{}^\circ) = \langle z \cdot xz(yz), \{x := \langle z' \cdot x', \{x' := s\}\rangle, y := \langle z'' \cdot z''\rangle\}\rangle \longrightarrow^+ \langle z \cdot \langle z' \cdot x', \{x' := x'\}\rangle z(\langle z'' \cdot z''\rangle z), \{x' := s\}\rangle$ (by (ι)) $\longrightarrow^+ \langle z \cdot x'z, \{x' := s\}\rangle$ (by (β)) $\longrightarrow_\eta s$ (by (η)). Otherwise:

$$\begin{array}{ll}
\mathcal{S}_{\ell+2}(\mathcal{K}_{\ell+1}(s), I_{\ell+1}{}^\circ) & \\
= \langle z \cdot \mathcal{S}_{\ell+1}(xz, yz), \{x := \langle z' \cdot \mathcal{K}_\ell(x'z'), \{x' := s\}\rangle, y := \langle z'' \cdot I_\ell{}^\circ\rangle\}\rangle & \\
\longrightarrow^+ \langle z \cdot \mathcal{S}_{\ell+1}(\langle z' \cdot \mathcal{K}_\ell(x'z'), \{x' := x'\}\rangle z, \langle z'' \cdot I_\ell{}^\circ\rangle z), \{x' := s\}\rangle & \text{(by } (\iota)) \\
\longrightarrow^+ \langle z \cdot \mathcal{S}_{\ell+1}(\mathcal{K}_\ell(x'z), I_\ell{}^\circ), \{x' := s\}\rangle & \text{(by } (\beta)) \\
\longrightarrow^*_\eta \langle z \cdot x'z, \{x' := s\}\rangle & \text{(by induction)} \\
\longrightarrow_\eta s & \text{(by } (\eta))
\end{array}$$

Item (v): We prove the claim by structural induction on s. If s is a variable, then $s^{*\circ} = s$. If s is an application $s_1 s_2$, then $(s_1 s_2)^{*\circ} = (S_0(s_1^*, s_2^*))^\circ = s_1{}^{*\circ} s_2{}^{*\circ} \approx s_1 s_2 = s$, by induction hypothesis. Finally, if s is a closure $\langle x \cdot t, \{x_1 := t_1, \ldots, x_k := t_k\}\rangle$, then $s^{*\circ} = (([x]t^*)[t_1^*/x_1, \ldots, t_k^*/x_k])^\circ$.

We first claim that: (g) for every SKInT-terms u, v_1, …, v_k, $(([x]u)[v_1/x_1, \ldots, v_k/x_k])^\circ \approx \langle x \cdot u^\circ, \{x_1 := v_1{}^\circ, \ldots, x_k := v_k{}^\circ\}\rangle$. The result will follow by applying (g) to $u = t^*$, $v_1 = t_1^*$, …, $v_k = t_k^*$ and using the induction hypothesis on t, t_1, …, t_k.

We prove (g) by structural induction on u. If $u = x$, then $(([x]u)[v_1/x_1, \ldots, v_k/x_k])^\circ = I_0{}^\circ = \langle x \cdot x\rangle$, and the right-hand side is $\langle x \cdot x\rangle\{x_1 := v_1{}^\circ, \ldots, x_k := v_k{}^\circ\}$, which reduces to $\langle x \cdot x\rangle$ in k (0) steps. If $u = x_i$ for some i, $1 \leq i \leq k$, then the left-hand side is $(K_0(v_i))^\circ = \langle x \cdot y, \{y := v_i{}^\circ\}\rangle$, while the right-hand side is $\langle x \cdot x_i, \{x_1 := v_1{}^\circ, \ldots, x_k := v_k{}^\circ\}\rangle$, which rewrites in $k-1$ (0) steps to $\langle x \cdot x_i, \{x_i := v_i{}^\circ\}\rangle$, the latter being α-equivalent to the left-hand side. If u is any other variable y, then the left-hand side is $(K_0(y))^\circ = \langle x \cdot z, \{z := y\}\rangle = \langle x \cdot y\rangle$, while the right-hand side is $\langle x \cdot y, \{x_1 := v_1{}^\circ, \ldots, x_k := v_k{}^\circ\}\rangle$, which rewrites to the latter in k (0) steps. If u is of the form I_ℓ, then the left-hand side is $I_{\ell+1}{}^\circ = \langle x \cdot I_\ell{}^\circ\rangle$, while the right-hand side is $\langle x \cdot I_\ell{}^\circ, \{x_1 := v_1{}^\circ, \ldots, x_k := v_k{}^\circ\}\rangle$, which rewrites to the latter in k (0) steps. If u is of the form $S_\ell(u_1, u_2)$, then the left-hand side is:

$$\begin{array}{ll}
(S_{\ell+1}(([x]u_1)[v_1/x_1,\ldots,v_k/x_k],([x]u_2)[v_1/x_1,\ldots,v_k/x_k]))^\circ & \\
= \mathcal{S}_{\ell+1}((([x]u_1)[v_1/x_1,\ldots,v_k/x_k])^\circ,(([x]u_2)[v_1/x_1,\ldots,v_k/x_k])^\circ) & \\
= \langle z\cdot\mathcal{S}_\ell(x'z,y'z),\{x' := (([x]u_1)[v_1/x_1,\ldots,v_k/x_k])^\circ, y' := (([x]u_2)[v_1/x_1,\ldots,v_k/x_k])^\circ\}\rangle & \\
\approx \langle z\cdot\mathcal{S}_\ell(x'z,y'z)\{x' := \langle x\cdot u_1{}^\circ,\{x_1 := v_1{}^\circ,\ldots,x_k := v_k{}^\circ\}\rangle, & \\
\qquad\qquad y' := \langle x\cdot u_2{}^\circ,\{x_1 := v_1{}^\circ,\ldots,x_k := v_k{}^\circ\}\rangle\}\rangle & \text{(by induction)} \\
\longrightarrow^+ \langle z\cdot\mathcal{S}_\ell(\langle x\cdot u_1{}^\circ,\{x_1 := x'_1,\ldots,x_k := x'_k\}\rangle z,\langle x\cdot u_2{}^\circ,\{x_1 := x''_1,\ldots,x_k := x''_k\}\rangle z) & \\
\qquad \{x'_1 := v_1{}^\circ,\ldots,x'_k := v_k{}^\circ, x''_1 := v_1{}^\circ,\ldots,x''_k := v_k{}^\circ\}\rangle & \text{(by }(\iota)\text{)} \\
\longrightarrow^+ \langle z\cdot\mathcal{S}_\ell(\langle x\cdot u_1{}^\circ,\{x_1 := x_1,\ldots,x_k := x_k\}\rangle z,\langle x\cdot u_2{}^\circ,\{x_1 := x_1,\ldots,x_k := x_k\}\rangle z) & \\
\qquad \{x_1 := v_1{}^\circ,\ldots,x_k := v_k{}^\circ\}\rangle & \text{(by (2))} \\
\longrightarrow^+ \langle x\cdot\mathcal{S}_\ell(u_1{}^\circ,u_2{}^\circ),\{x_1 := v_1{}^\circ,\ldots,x_k := v_k{}^\circ\}\rangle & \text{(by }(\beta)\text{)} \\
= \langle x\cdot u^\circ,\{x_1 := v_1{}^\circ,\ldots,x_k := v_k{}^\circ\}\rangle &
\end{array}$$

which is the desired right-hand side. And if u is of the $K_\ell(u_1)$, then the left-hand side is:

$$\begin{array}{ll}
((K_{\ell+1}([x]u_1))[v_1/x_1,\ldots,v_k/x_k])^\circ & \\
= \mathcal{K}_{\ell+1}((([x]u_1)[v_1/x_1,\ldots,v_k/x_k])^\circ) & \\
= \langle z\cdot\mathcal{K}_\ell(x'z),\{x' := (([x]u_1)[v_1/x_1,\ldots,v_k/x_k])^\circ\}\rangle & \\
\approx \langle z\cdot\mathcal{K}_\ell(x'z),\{x' := \langle x\cdot u_1{}^\circ,\{x_1 := v_1{}^\circ,\ldots,x_k := v_k{}^\circ\}\rangle\}\rangle & \text{(by induction)} \\
\longrightarrow \langle z\cdot\mathcal{K}_\ell(\langle x\cdot u_1{}^\circ,\{x_1 := x_1,\ldots,x_k := x_k\}\rangle z),\{x_1 := v_1{}^\circ,\ldots,x_k := v_k{}^\circ\}\rangle & \text{(by }(\iota)\text{)} \\
\longrightarrow \langle x\cdot\mathcal{K}_\ell(u_1{}^\circ),\{x_1 := v_1{}^\circ,\ldots,x_k := v_k{}^\circ\}\rangle & \text{(by }(\beta)\text{)} \\
= \langle x\cdot u^\circ,\{x_1 := v_1{}^\circ,\ldots,x_k := v_k{}^\circ\}\rangle &
\end{array}$$

and this is the desired right-hand side.

Note finally that saying as in (v) that $s^{*\circ} \approx s$ can be refined into saying that in fact $s^{*\circ}$ rewrites to s, up to some applications of (0) in the reverse order; alternatively, if s contains no (0)-redex (garbage collection has already been called on s), then $s^{*\circ} \longrightarrow^* s$.

Item (vi): We prove the claim by structural induction on u. Write $\longrightarrow_\eta$ for the reduction relation of SKInT$_\eta$. If u is a variable, then $u^{\circ *} = u$.

If u is of the form I_ℓ, then we show that $u^{\circ *} = u$ by induction on ℓ. If $\ell = 0$, $I_0{}^{\circ *} = \langle z\cdot z\rangle^* = I_0 = u$. Otherwise, $I_{\ell+1}{}^{\circ *} = \langle z\cdot I_\ell{}^\circ\rangle^* = [z]I_\ell{}^{\circ *} = [z]I_\ell$ (by induction) $= I_{\ell+1} = u$.

If u is of the form $S_\ell(u_1, u_2)$, then we first show that: (h) $\mathcal{S}_\ell(s,t)^* \longrightarrow_\eta^* S_\ell(s^*, t^*)$. Then $u^{\circ *} \longrightarrow_\eta^* u$ follows from (h) and the induction hypothesis. In turn, we show (h) by induction on ℓ. If $\ell = 0$, then $\mathcal{S}_0(s,t)^* = (st)^* = S_0(s^*, t^*)$. Otherwise, $\mathcal{S}_{\ell+1}(s,t)^* = \langle z\cdot\mathcal{S}_\ell(xz,yz),\{x := s, y := t\}\rangle^* = ([z]\mathcal{S}_\ell(xz,yz)^*)[s^*/x,t^*/y] \longrightarrow_\eta^* ([z]S_\ell((xz)^*,(yz)^*))[s^*/x,t^*/y]$ (by induction hypothesis) $= ([z]S_\ell(S_0(x,z),S_0(y,z)))[s^*/x,t^*/y] = S_{\ell+1}(S_1(K_0(s^*),I_0),S_1(K_0(t^*),I_0)) \longrightarrow_\eta^+ S_{\ell+1}(s^*,t^*)$ (by the η-rule).

If u is of the form $K_\ell(u_1)$, then we first show that: (i) $\mathcal{K}_\ell(s)^* \longrightarrow_\eta^* K_\ell(s^*)$. Then $u^{\circ *} \longrightarrow_\eta^* u$ follows from (h) and the induction hypothesis. In turn, we show (i) by induction on ℓ. If $\ell = 0$, then $\mathcal{K}_0(s)^* = \langle z\cdot x,\{x := s\}\rangle^* = ([z]x)[s^*/x] = K_0(s^*)$. Otherwise, $\mathcal{K}_{\ell+1}(s)^* = \langle z\cdot\mathcal{K}_\ell(xz),\{x := s\}\rangle^* = ([z]\mathcal{K}_\ell(xz)^*)[s^*/x] \longrightarrow_\eta^* ([z]K_\ell(S_0(x,z)))[s^*/x]$ (by induction hypothesis) $= K_{\ell+1}(S_1(K_0(s^*),I_0)) \longrightarrow_\eta K_{\ell+1}(s^*)$ (by the η rule).

Notice that we have actually used only the η-rule, and no other.

Item (vii): first, define the degree $d(u)$ of a SKInT-term u by $d(x) = 0$, $d(I_\ell) = \ell+1$, $d(S_\ell(u_1,u_2)) = \ell$, $d(K_\ell(u_1)) = \ell+1$. Notice that $d(([x]u)[v_1/x_1,\ldots,v_k/x_k]) = d(u)+1$ for every terms u, v_1, …, v_k.

We first claim that: (j) if u, v_1, …, v_k are normal SKInT-terms, and v_1, …, v_k are of degree 0, then $([x]u)[v_1/x_1,\ldots,v_k/x_k]$ is normal as well. This is proved by structural induction on u. The only difficult case is when u is some variable x_i, $1 \leq i \leq n$ ($x_i \neq x$). Then $([x]u)[v_1/x_1,\ldots,v_k/x_k] = K_0(v_i)$ can only rewrite at the top of the term; but this is impossible because v_i, being of degree 0, is a variable or has head function symbol S_0, and in each case no rule applies. If u is some other variable, then $([x]u)[v_1/x_1,\ldots,v_k/x_k]$ is I_0 if $u = x$ or $K_0(u)$ otherwise, and is therefore normal. If u is of the form $S_\ell(u_1,u_2)$, then observe (by inspection of the rules of SKInT) that such a term is normal if and only if u_1 and u_2 are normal, and $d(u_1) \leq \ell$; then

$([x]u)[v_1/x_1,\dots,v_k/x_k] = S_{\ell+1}(([x]u_1)[v_1/x_1,\dots,v_k/x_k], ([x]u_2)[v_1/x_1,\dots,v_k/x_k])$ is normal, by induction hypothesis, and since $d(([x]u_1)[v_1/x_1,\dots,v_k/x_k]) = d(u_1)+1 \leq \ell+1$. And if u is of the form $K_\ell(u_1)$, the argument is similar, noticing that such a term is normal if and only if u_1 is and $d(u_1) \leq \ell$.

Now we prove that for every λ_{clos}-normal term s, s^* is SKInT-normal. This is by structural induction on s. If s is a variable, then this is clear. If s is an application $s_1 s_2$, then s_1 is a variable or an application, so that s_1^* is of degree 0 in each case: then $s^* = S_0(s_1^*, s_2^*)$, where s_1^* and s_2^* are normal by induction hypothesis; since s_1^* is of degree 0, a simple check shows that there is no redex at the top of s^* either, i.e. s^* is normal (in particular, s_1^* is not I_0, whose degree is 1, not 0). If s is a closure $\langle x \cdot t, \{x_1 := t_1, \dots, x_k := t_k\}\rangle$, then $s^* = ([x]t^*)[t_1^*/x_1,\dots,t_k^*/x_k]$; now, t^*, t_1^*, …, t_k^* are normal by induction hypothesis, and since s is normal (in particular, rule (ι) does not apply), t_1, …, t_k are not closures; so t_1^*, …, t_k^* are of degree 0: then (j) applies, and proves the claim.

In the presence of η-rule (both in λ_{clos} and SKInT), we adapt the proof above by replacing (j) by claim (j'): if u, v_1, …, v_k are SKInT_η-normal SKInT-terms, and v_1, …, v_k are of degree 0, *and* u is not of the form $S_0(z,x)$ for some variable $z \neq x$, then $([x]u)[v_1/x_1,\dots,v_k/x_k]$ is SKInT_η-normal as well. Indeed, the only case that changes is when u is of the form $S_\ell(u_1,u_2)$: such a term is SKInT_η-normal if and only if u_1 and u_2 are, and $d(u_1) \leq \ell$, *and* it is not the case that u_1 is of the form $K_{\ell-1}(u_3)$ for any u_3 and $u_2 = I_{\ell-1}$. Now $([x]u)[v_1/x_1,\dots,v_k/x_k] = S_{\ell+1}(([x]u_1)[v_1/x_1,\dots,v_k/x_k], ([x]u_2)[v_1/x_1,\dots,v_k/x_k])$, and therefore it is normal if and only if $([x]u_1)[v_1/x_1,\dots,v_k/x_k]$ and $([x]u_2)[v_1/x_1,\dots,v_k/x_k]$ are (by induction), $d(([x]u_1)[v_1/x_1,\dots,v_k/x_k]) \leq \ell+1$ (since the left-hand side equals $d(u_1)+1$), and it is not the case that $([x]u_1)[v_1/x_1,\dots,v_k/x_k]$ is of the form $K_\ell(u_4)$ for any u_4 and $([x]u_2)[v_1/x_1,\dots,v_k/x_k] = I_\ell$. But the latter happens (by inspection) if and only if either $\ell \geq 1$, u_1 is of the form $K_{\ell-1}(u_4)$ and $u_2 = I_{\ell-1}$ (which is excluded by assumption), or $\ell = 0$, u_1 is a variable z other than z, and $u_2 = x$: but then u would be $S_0(z,x)$, which is exactly what we have forbidden.

We then use (j') instead of (j) to show by structural induction on s that for every $\lambda_{\text{clos}\,\eta}$-normal term s, s^* is SKInT_η-normal. The only new case is when s is a closure $\langle x \cdot t, \{x_1 := t_1, \dots, x_k := t_k\}\rangle$. Then $s^* = ([x]t^*)[t_1^*/x_1,\dots,t_k^*/x_k]$; now, t^*, t_1^*, …, t_k^* are normal by induction hypothesis, and since s is normal (in particular, rule (ι) does not apply), t_1, …, t_k are not closures; so t_1^*, …, t_k^* are of degree 0; moreover, since rule (η) does not apply, t is not of the form zx for any variable $z \neq x$, so t^* is not of the form $S_0(z,x)$: then (j') applies, and proves the claim.

E Solvability

For convenience, we shall consider *partial spines* P, which are sequences of operators S_ℓ or K_ℓ, $\ell \geq 0$. A spine is just the concatenation PI_ℓ or Px for some partial spine P. In general, we define $P(u)[u_1,\dots,u_n]$ as follows (letting ϵ denote the empty partial spine):

$$\begin{array}{ll} \epsilon(u)[] & = u \\ (S_\ell P')(u)[u_1,\dots,u_n] & = S_\ell(P'(u)[u_1,\dots,u_{n-1}], u_n) \\ (K_\ell P')(u)[u_1,\dots,u_n] & = K_\ell(P'(u)[u_1,\dots,u_n]) \end{array}$$

We claim that: (a) if u_1 rewrites to u_2 in one spine βI-step, and u_1 rewrites to u_3 in one ΣT-step, then u_2 and u_3 have a common reduct u_4, such that u_3 rewrites to u_4 by some ΣT-steps followed by one spine βI-step, and u_2 rewrites to u_4 by zero or one ΣT-step. In short, if $u_1 \longrightarrow^{s\beta I} u_2$ and $u_1 \longrightarrow^{\Sigma T} u_3$, then $u_2 \longrightarrow^{\Sigma T^{\{0,1\}}} u_4$ and $u_3 \longrightarrow^{\Sigma T^*\, s\beta I} u_4$ for some term u_4.

First, (a) is clear when the ΣT-redex that we contract to get from u_1 to u_3 is not on the spine. In this case indeed, u_1 is of the form $(PS_\ell I_\ell)[v_1,\dots,v_n]$, $u_3 = (PS_\ell I_\ell)[v_1,\dots,v_{i-1},v_i',v_{i+1},\dots,v_n]$, with $v_i \longrightarrow^{\Sigma T} v_i'$; then $u_2 = P(v_1)[v_2,\dots,v_n]$ clearly rewrites in one ΣT-step to:

$$u_4 =_{\text{df}} \begin{cases} P(v_1')[v_2,\dots,v_n] & \text{if } i = 1 \\ P(v_1)[v_2,\dots,v_{i-1},v_i',v_{i+1},\dots,v_n] & \text{if } i \geq 2 \end{cases}$$

It is then clear that u_3 rewrites to u_4 in one spine βI-step.

If the ΣT-redex in u_1 is disjoint from the spine βI-redex, (a) is clear again. So it remains to examine the cases where the ΣT-redex and the spine βI superpose. There are two cases:

- If u_1 is of the form $(PS_\ell S_\mathcal{L} I_\mathcal{L})[v_1, v_2, \ldots, v_n]$, with $0 \leq \ell < \mathcal{L}$, $u_2 = (PS_\ell)(v_1)[v_2, \ldots, v_n]$, and $u_3 = (PS_{\mathcal{L}-1} S_\ell I_\mathcal{L})[v_2, S_\ell(v_1, v_2), v_3, \ldots, v_n]$. So u_3 rewrites to $(PS_{\mathcal{L}-1} I_{\mathcal{L}-1})[S_\ell(v_1, v_2), v_3, \ldots, v_n]$ (by ΣT), then to $P(S_\ell(v_1, v_2))[v_3, \ldots, v_n] = (PS_\ell)(v_1)[v_2, v_3, \ldots, v_n] = u_2$ (by βI).

- If u_1 is of the form $(PK_\ell S_\mathcal{L} I_\mathcal{L})[v_1, v_2, \ldots, v_n]$, then $u_2 = (PK_\ell)(v_1)[v_2, \ldots, v_n]$ and $u_3 = (PS_{\mathcal{L}+1} K_\ell I_\mathcal{L})[K_\ell(v_1), v_2, \ldots, v_n]$. Then u_3 rewrites to $(PS_{\mathcal{L}+1} I_{\mathcal{L}+1})[K_\ell(v_1), v_2, \ldots, v_n]$ (by ΣT), then to $P(K_\ell(v_1))[v_2, \ldots, v_n] = u_2$ by βI.

This proves (a).

We now claim that: (b) if $u_1 \longrightarrow^{s\beta I} u_2$ and $u_1 \longrightarrow^{\Sigma T^*} u_3$, then $u_2 \longrightarrow^{\Sigma T^*} u_4$ and $u_3 \longrightarrow^{\Sigma T^*\, s\beta I} u_4$ for some term u_4. (Compared to (a), we have just replaced ΣT by ΣT^*.) This is by induction on the length $\nu(u_1)$ of the longest ΣT-reduction starting from u_1. If $u_1 = u_3$, then we take $u_4 =_{\mathrm{df}} u_2$. Otherwise, the proof is by:

$$
\begin{tikzcd}
u_1 \arrow[rr, "s\beta I"] \arrow[d, "\Sigma T"'] \arrow[drr, phantom, "\text{by (a)}"] & & u_2 \arrow[d, "\Sigma T^{\{0,1\}}"] \\
\cdot \arrow[r, "\Sigma T^*"] \arrow[d, "\Sigma T^*"'] \arrow[dr, phantom, "\text{by confluence of } \Sigma T"] & \cdot \arrow[r, "s\beta I"] \arrow[d, "\Sigma T^*"] \arrow[dr, phantom, "\text{by induction}"] & \cdot \arrow[d, "\Sigma T^*"] \\
u_3 \arrow[r, "\Sigma T^*"'] & \cdot \arrow[r, "\Sigma T^*\ s\beta I"'] & u_4
\end{tikzcd}
$$

It follows from (b) that $u_1 \longrightarrow^{s\beta I} u_2$ implies $\Sigma T(u_1) \rightsquigarrow \Sigma T(u_2)$. Moreover, if u_1 rewrites to u_2 by any other spine reduction, then this is a ΣT-reduction, hence $\Sigma T(u_1) = \Sigma T(u_2)$. In other words, if $u_1 \longrightarrow^{s*} u_2$ using βI n times, then $\Sigma T(u_1) \rightsquigarrow^* \Sigma T(u_2)$ in exactly n steps, as claimed.

F Normalization properties of $\lambda_{\oplus\epsilon}$

We first show: (a) If $\Gamma \vdash t : \tau$ in $\mathcal{S}\omega$, then t is solvable (i.e., every head-reduction terminates). Notice that the erasure $|t|$ of t is solvable in the λ-calculus (with β-reduction), because $|t|$ has a type in $\mathcal{S}\omega$ [Say97]. But then, if $t = t_0 \longrightarrow t_1 \longrightarrow \ldots \longrightarrow t_k \longrightarrow \ldots$ is any head-reduction starting from t, then $|t_0| \longrightarrow^= |t_1| \longrightarrow^= \ldots \longrightarrow^= |t_k| \longrightarrow^= \ldots$ is a head-reduction in the λ-calculus by definition (where $\longrightarrow^=$ denotes zero or one step of $\longrightarrow$), and is therefore finite. So, if the original reduction $t_0 \longrightarrow t_1 \longrightarrow \ldots \longrightarrow t_k \longrightarrow \ldots$ was infinite, then for k high enough, we would have $|t_k| = |t_{k+1}| = \ldots = |t_j| = \ldots$. But now, the reduction $t_k \longrightarrow t_{k+1} \longrightarrow \ldots \longrightarrow t_j \longrightarrow \ldots$ takes place entirely inside the subsystem defined by rules (ϵ), $(\oplus-)$, $(\oplus)$ and (η). But the latter system terminates, which leads to a contradiction. (Indeed, map $\lambda_{\oplus\epsilon}$-terms t to first-order terms $f(t)$, by: $f(x) =_{\mathrm{df}} x$, $f(\lambda x \cdot t) =_{\mathrm{df}} f(t)[c/x]$, where c is a fixed constant, $f(st) =_{\mathrm{df}} @(s, t)$, $f(\epsilon s) =_{\mathrm{df}} \epsilon f(s)$, $f(s \oplus t) =_{\mathrm{df}} f(s) \oplus f(t)$, which we order by the well-founded lexicographic path ordering on an arbitrary precedence [Der87].)

We now show: (b) If $\Gamma^- \vdash t : \tau^+$ in $\mathcal{S}\omega$, with Γ^- definite negative and τ^+ definite positive, then t is weakly normalizing. This is trivial: taking the normal form of t with respect to rules (ϵ) and $(\oplus-)$ (which terminates, by the remark above) clearly yields $|t|$, a λ-term with the same typing as t. By standard results [Say97], $|t|$ normalizes weakly in the λ-calculus, hence also in the $\lambda_{\oplus\epsilon}$-calculus (in which the λ-calculus is contained) to some normal form t'. It follows that $t \longrightarrow^* |t| \longrightarrow^* t'$ is a normalizing reduction in $\lambda_{\oplus\epsilon}$.

But we wish to show more, namely that every standard reduction in $\lambda_{\oplus\epsilon}$ terminates under the assumptions of (b). Indeed, every standard β-reduction starting from $|t|$ terminates [Say97], so there cannot be infinite sequences $t_i \longrightarrow^= t_{i+1} \longrightarrow^= \ldots \longrightarrow^= t_j$ where $t_i = t_j$, since (ϵ), $(\oplus-)$, $(\oplus)$, (η) defines a terminating first-order rewrite system (as in the solvability case).

Finally, we show: (c) If $\Gamma \vdash t : \tau$ in $\mathcal{S}$, then t is strongly normalizing. Again, $|t|$ is a λ-term with the same typing as t, and therefore normalizes strongly [Say97] in the λ-calculus with β-reduction. Let therefore $\nu(t)$ denote the length of the longest normalizing sequence in the λ-calculus starting from $|t|$. Let also $s(t)$ denote the size of t, and, if t is strongly normalizing in $\lambda_{\oplus\epsilon}$, let $\nu_\oplus(t)$ denote the length of its longest normalizing sequence in $\lambda_{\oplus\epsilon}$. Using this, assume that: $(*)$ every $\lambda_{\oplus\epsilon}$-term s such that $\nu(|s'|) < n$ for every subterm s' of s is strongly normalizing. Then we show that: (d) every $\lambda_{\oplus\epsilon}$-term s such that $\nu(|s'|) < n+1$ for every subterm s' of s is strongly normalizing as well. We show this by structural induction on s; the various cases to check are:

- Every variable is strongly normalizing in $\lambda_{\oplus\epsilon}$: trivial.
- If t and t' are strongly normalizing $\lambda_{\oplus\epsilon}$-terms, and $\nu(|tt'|) < n+1$, then we show that tt' is strongly normalizing in $\lambda_{\oplus\epsilon}$. This is by induction on $\nu_\oplus(t) + \nu_\oplus(t')$: if tt' is normal, this is clear, otherwise given a non-empty reduction starting from tt', we observe that it is finite: if the first redex in the sequence is at the top, then tt' (β)-reduces to some term s such that $\nu(|s|) < nu(|tt'|)$, so $\nu(|s|) < n$ and by assumption $(*)$ the reduction is finite; otherwise, the reduction starts by reducing a redex in t or in t': if the contracted redex is in t, then tt' rewrites to $t''t'$ with $\nu(|t''t'|) \leq \nu(|tt'|) < n+1$, and the induction hypothesis applies, since $\nu_\oplus(t'') + \nu_\oplus(t') < \nu_\oplus(t) + \nu_\oplus(t')$; similarly if the reduction starts by reducing a redex in t'.
- If t is strongly normalizing in $\lambda_{\oplus\epsilon}$, then so is $\lambda x \cdot t$. This is by induction on $\nu_\oplus(t)$. If $\lambda x \cdot t$ is normal, then the result is trivial. Otherwise, the first step is either a reduction step in t, and we apply the induction hypothesis; or the first step is an (η)-step at the top, i.e. t is of the form $t'x$ with x not free in t', but then the reduction terminates again, since t', as a subterm of t, is strongly normalizing.
- If t is strongly normalizing in $\lambda_{\oplus\epsilon}$, then so is ϵt. Indeed, given any reduction R starting from t, either it lies entirely inside t (then the result is clear), or eventually (ϵ) gets applies at the top, i.e., $\epsilon t = \epsilon t_0 \longrightarrow \epsilon t_1 \longrightarrow \ldots \longrightarrow \epsilon t_k \longrightarrow^{(\epsilon)} t_k \longrightarrow t_{k+1} \longrightarrow \ldots$: but then the reduction $t_0 \longrightarrow t_1 \longrightarrow \ldots \longrightarrow t_k \longrightarrow t_{k+1} \longrightarrow \ldots$ is finite by assumption, hence R is finite as well.
- If t and t' are strongly normalizing in $\lambda_{\oplus\epsilon}$, then so is $t \oplus t'$. We show this by induction on $(\nu_\oplus(t), \nu_\oplus(t'))$ ordered lexicographically. Consider any reduction starting from R. If it is empty, then it is clearly finite. If it starts by reducing a redex in t or in t', then this follows from the induction hypothesis. If R starts by rule $(\oplus-)$ at the top, then the rest of R is a reduction inside t', and is therefore finite by assumption. If R starts by rule $(\oplus)$ at the top, then t was of the form $t_1 \oplus t_2$, and R first rewrites $t \oplus t'$ to $t_1 \oplus (t_2 \oplus t')$. But then $\nu_\oplus(t_1) < \nu_\oplus(t)$, since there is a reduction of t that first reduces t_1 by its longest reduction (of length $\nu_\oplus(t_1)$) then applies $(\oplus-)$ at the end: so again the induction hypothesis applies.

This proves (d) under the assumption $(*)$. It follows by induction on n that every $\lambda_{\oplus\epsilon}$-term s such that $|s'|$ is strongly normalizing for every subterm s' of s is itself strongly normalizing. Now (c) obtains, since if s is typable in system S, then so are all of its subterms s', and therefore $|s'|$ is typable in system $\mathcal{S}$, hence strongly normalizing.

G Interpretation of SKInT-reductions in $\lambda_{\oplus\epsilon}$

First, we notice that: (a) if $s_0 \longrightarrow^* s'_0, \ldots, s_{n-1} \longrightarrow^* s'_{n-1}$ in SKInT, then $[\![u]\!]_\oplus(s_0, \ldots, s_{n-1}) \longrightarrow^* [\![u]\!]_\oplus(s'_0, \ldots, s'_{n-1})$ in $\lambda_{\oplus\epsilon}$; and that, if moreover $s_i \longrightarrow^+ s'_i$ for some i, $0 \leq i < n$, then $[\![u]\!]_\oplus(s_0, \ldots, s_{n-1}) \longrightarrow^+ [\![u]\!]_\oplus(s'_0, \ldots, s'_{n-1})$. This is by a straightforward structural induction on u (or rather the chosen typing derivation for u). The first part of the claim is in fact almost trivial, while the second part depends on the fact that no s_i is dropped on the right-hand side of the various clauses defining $[\![u]\!]_\oplus(s_0, \ldots, s_{n-1})$.

Then, notice that: (b) $[\![u]\!]_\oplus(s_0, \ldots, s_{n-1})\sigma = [\![u]\!]_\oplus(s_0\sigma, \ldots, s_{n-1}\sigma)$ for any substitution σ whose domain does not intersect the set of free variables of u. This is straightforward.

Then, notice that: (c) $[\![u]\!]_\oplus(s_0,\dots,s_{n-1})s_n\dots s_{m-1} \longrightarrow^* [\![u]\!]_\oplus(s_0,\dots,s_{n-1},s_n,\dots,s_{m-1})$ whenever $m \geq n$. This is by structural induction on u, and we only have to consider the cases $m > n$, since $m = n$ is trivial. If u is a variable x, this is clear. If $u = I_\ell$, then we have three cases:

- If $m > n \geq \dim u = \ell+1$, then we have $[\![u]\!]_\oplus(s_0,\dots,s_{n-1})s_n\dots s_{m-1} = \epsilon(s_0 \oplus \dots \oplus s_{\ell-1} \oplus s_\ell)s_{\ell+1}\dots s_{n-1}s_n\dots s_{m-1} = [\![u]\!]_\oplus(s_0,\dots,s_{n-1},s_n,\dots,s_{m-1})$;
- if $m \geq \dim u > n$, then $[\![u]\!]_\oplus(s_0,\dots,s_{n-1})s_n\dots s_{m-1} = (\lambda x_n \cdot \dots \cdot \lambda x_\ell \cdot \epsilon(s_0 \oplus \dots \oplus s_{n-1} \oplus x_n \oplus \dots \oplus x_{\ell-1} \oplus x_\ell))s_n\dots s_{m-1} \longrightarrow^* \epsilon(s_0 \oplus \dots \oplus s_{n-1} \oplus s_n \oplus \dots \oplus s_{\ell-1} \oplus s_\ell)s_{\ell+1}\dots s_{m-1}$ (by (β)) $= [\![u]\!]_\oplus(s_0,\dots,s_{n-1},s_n,\dots,s_{m-1})$;
- if $\dim u = \ell+1 > m > n$, then $[\![u]\!]_\oplus(s_0,\dots,s_{n-1})s_n\dots s_{m-1} = (\lambda x_n \cdot \dots \cdot \lambda x_\ell \cdot \epsilon(s_0 \oplus \dots \oplus s_{n-1} \oplus x_n \oplus \dots \oplus x_{\ell-1} \oplus x_\ell))s_n\dots s_{m-1} \longrightarrow^* \lambda x_m \cdot \dots \lambda x_\ell \cdot \epsilon(s_0 \oplus \dots \oplus s_{n-1} \oplus s_n \oplus \dots \oplus s_{\ell-1} \oplus s_\ell)$ (by (β)) $= [\![u]\!]_\oplus(s_0,\dots,s_\ell)$.

If u is of the form $K_\ell(v)$, then let $q =_{\mathrm{df}} \dim u = \max(\ell+2, \dim v + 1)$:

- If $m > n \geq q$, then $[\![u]\!]_\oplus(s_0,\dots,s_{n-1})s_n\dots s_{m-1} = [\![v]\!]_\oplus(s_0,\dots,s_{\ell-1},s_\ell \oplus s_{\ell+1},s_{\ell+2},\dots,s_{n-1})$ $s_n\dots s_{m-1} \longrightarrow^* [\![v]\!]_\oplus(s_0,\dots,s_{\ell-1},s_\ell \oplus s_{\ell+1},s_{\ell+2},\dots,s_{n-1},s_n\dots s_{m-1})$ (by induction) $= [\![u]\!]_\oplus(s_0,\dots,s_{m-1})$;
- if $m \geq q > n$, then we have $[\![u]\!]_\oplus(s_0,\dots,s_{n-1})s_n\dots s_{m-1} = (\lambda x_n \cdot \dots \cdot \lambda x_{q-1} \cdot [\![u]\!]_\oplus(s_0,\dots,s_{n-1},x_n,\dots,x_{q-1}))s_n\dots s_{m-1} \longrightarrow^* [\![u]\!]_\oplus(s_0,\dots,s_{n-1},x_n,\dots,x_{q-1})$ $[s_n/x_n,\dots,s_{q-1}/x_{q-1}]s_q\dots s_{m-1}$ (by (β)) $= [\![u]\!]_\oplus(s_0,\dots,s_{n-1},s_n,\dots,s_{q-1})s_q\dots s_{m-1}$ (by (b)) $= [\![v]\!]_\oplus(s_0,\dots,s_{\ell-1},s_\ell \oplus s_{\ell+1},s_{\ell+2},\dots,s_{q-1})s_q\dots s_{m-1}$ (by definition, since $q \geq \dim u$) $\longrightarrow^*$ $[\![v]\!]_\oplus(s_0,\dots,s_{\ell-1},s_\ell \oplus s_{\ell+1},s_{\ell+2},\dots,s_{q-1},s_q,\dots,s_{m-1}) = [\![u]\!]_\oplus(s_0,\dots,s_{m-1})$;
- if $q > m > n$, then $[\![u]\!]_\oplus(s_0,\dots,s_{n-1})s_n\dots s_{m-1} = (\lambda x_n \cdot \dots \lambda x_{q-1} \cdot [\![u]\!]_\oplus(s_0,\dots,s_{n-1},x_n,\dots,x_{q-1}))$ $s_n\dots s_{m-1} \longrightarrow^* \lambda x_m \cdot \dots \cdot \lambda x_{q-1} \cdot [\![u]\!]_\oplus(s_0,\dots,s_{m-1},x_m,\dots,x_{q-1})$ (by (β), using (b)) $=$ $[\![u]\!]_\oplus(s_0,\dots,s_{m-1})$ (because $m < \dim u$).

And if u is of the form $S_\ell(v,w)$, then let $q =_{\mathrm{df}} \dim u = \max(\ell, \dim v - 1)$:

- If $m > n \geq q$, then $[\![u]\!]_\oplus(s_0,\dots,s_{n-1})s_n\dots s_{m-1} = [\![v]\!]_\oplus(s_0,\dots,s_{\ell-1},[\![w]\!]_\oplus(s_0,\dots,s_{\ell-1}),s_\ell,\dots,s_{n-1})$ $s_n\dots s_{m-1} \longrightarrow^* [\![v]\!]_\oplus(s_0,\dots,s_{\ell-1},[\![w]\!]_\oplus(s_0,\dots,s_{\ell-1}),s_\ell,\dots,s_{n-1},s_n\dots s_{m-1})$ (by induction) $=$ $[\![u]\!]_\oplus(s_0,\dots,s_{m-1})$;
- if $m \geq q > n$, then $[\![u]\!]_\oplus(s_0,\dots,s_{n-1})s_n\dots s_{m-1} = (\lambda x_n \cdot \dots \cdot \lambda x_{q-1} \cdot$ $[\![u]\!]_\oplus(s_0,\dots,s_{n-1},x_n,\dots,x_{q-1}))s_{n-1}\dots s_{m-1} \longrightarrow^* [\![u]\!]_\oplus(s_0,\dots,s_{q-1})s_q\dots s_{m-1}$ (by (β), using (b)) $= [\![v]\!]_\oplus(s_0,\dots,s_{\ell-1},[\![w]\!]_\oplus(s_0,\dots,s_{\ell-1}),s_\ell,\dots,s_{q-1})s_q\dots s_{m-1} \longrightarrow^*$ $[\![v]\!]_\oplus(s_0,\dots,s_{\ell-1},[\![w]\!]_\oplus(s_0,\dots,s_{\ell-1}),s_\ell,\dots,s_{q-1},s_q\dots s_{m-1})$ (by induction) $= [\![u]\!]_\oplus(s_0,\dots,s_{m-1})$;
- if $q > m > n$, then $[\![u]\!]_\oplus(s_0,\dots,s_{n-1})s_n\dots s_{m-1} = (\lambda x_n \cdot \dots \cdot \lambda x_{q-1} \cdot$ $[\![u]\!]_\oplus(s_0,\dots,s_{n-1},x_n,\dots,x_{q-1}))s_n\dots s_{m-1} \longrightarrow^* \lambda x_m \cdot \dots \cdot \lambda x_{q-1} \cdot [\![u]\!]_\oplus(s_0,\dots,s_{m-1},x_m,\dots,x_{q-1})$ (by (β), using (b)) $= [\![u]\!]_\oplus(s_0,\dots,s_{m-1})$ (because $m < \dim u$).

Then we also have: (d) for any fresh variables x_n, …, x_{m-1} ($m \geq n$), $\lambda x_n \cdot \dots \cdot \lambda x_{m-1} \cdot$ $[\![u]\!]_\oplus(s_0,\dots,s_{n-1},x_n,\dots,x_{m-1}) \longrightarrow^* [\![u]\!]_\oplus(s_0,\dots,s_{n-1})$. This is by structural induction on u. If $u = I_\ell$, then we have three cases:

- If $m > n \geq \ell+1$, then $\lambda x_n \cdot \dots \cdot \lambda x_{m-1} \cdot [\![u]\!]_\oplus(s_0,\dots,s_{n-1},x_n,\dots,x_{m-1}) = \lambda x_n \cdot \dots \cdot \lambda x_{m-1} \cdot \epsilon(s_0 \oplus \dots \oplus s_\ell)$ $s_{\ell+1}\dots s_{n-1}x_n\dots x_{m-1} \longrightarrow^* \epsilon(s_0 \oplus \dots \oplus s_\ell)s_{\ell+1}\dots s_{n-1}$ (by (η)) $= [\![u]\!]_\oplus(s_0,\dots,s_{n-1})$;
- if $m \geq \ell+1 > n$, then $\lambda x_n \cdot \dots \cdot \lambda x_{m-1} \cdot [\![u]\!]_\oplus(s_0,\dots,s_{n-1},x_n,\dots,x_{m-1}) = \lambda x_n \cdot \dots \cdot \lambda x_{m-1} \cdot$ $\epsilon(s_0 \oplus \dots \oplus s_{n-1} \oplus x_n \oplus \dots \oplus x_\ell)x_{\ell+1}\dots x_{m-1} \longrightarrow^* \lambda x_n \cdot \dots \cdot \lambda x_\ell \cdot \epsilon(s_0 \oplus \dots \oplus s_{n-1} \oplus x_n \oplus \dots \oplus x_\ell)$ (by (η)) $= [\![u]\!]_\oplus(s_0,\dots,s_{n-1})$;
- if $\ell+1 > m > n$, then $\lambda x_n \cdot \dots \cdot \lambda x_{m-1} \cdot [\![u]\!]_\oplus(s_0,\dots,s_{n-1},x_n,\dots,x_{m-1}) = \lambda x_n \cdot \dots \cdot \lambda x_{m-1} \cdot \lambda x_m \cdot \dots \cdot \lambda x_\ell \cdot$ $\epsilon(s_0 \oplus \dots \oplus s_{n-1} \oplus x_n \oplus \dots \oplus x_\ell) = [\![u]\!]_\oplus(s_0,\dots,s_{n-1})$.

If u is of the form $K_\ell(v)$, then let $q =_{\rm df} \dim u = \max(\ell+2, \dim v + 1)$; we have three cases:

- If $m > n \geq q$, then $\lambda x_n \cdot \ldots \cdot \lambda x_{m-1} \cdot [\![u]\!]_\oplus(s_0, \ldots, s_{n-1}, x_n, \ldots, x_{m-1}) = \lambda x_n \cdot \ldots \cdot \lambda x_{m-1} \cdot [\![v]\!]_\oplus(s_0, \ldots, s_{\ell-1}, s_\ell \oplus s_{\ell+1}, s_{\ell+2}, \ldots, s_{n-1}, x_n, \ldots, x_{m-1})$ $\longrightarrow^* [\![v]\!]_\oplus(s_0, \ldots, s_{\ell-1}, s_\ell \oplus s_{\ell+1}, s_{\ell+2}, \ldots, s_{n-1})$ (by induction) $= [\![u]\!]_\oplus(s_0, \ldots, s_{n-1})$;
- if $m \geq q > n$, then $\lambda x_n \cdot \ldots \cdot \lambda x_{m-1} \cdot [\![u]\!]_\oplus(s_0, \ldots, s_{n-1}, x_n, \ldots, x_{m-1}) = \lambda s_n \cdot \ldots \cdot \lambda s_{m-1} \cdot [\![u]\!]_\oplus(s_0, \ldots, s_{m-1})$ (by α-renaming: this will save us some tedious and useless case splitting) $= \lambda s_n \cdot \ldots \cdot \lambda s_{m-1} \cdot [\![v]\!]_\oplus(s_0, \ldots, s_{\ell-1}, s_\ell \oplus s_{\ell+1}, s_{\ell+2}, \ldots, s_{m-1}) \longrightarrow^* \lambda s_n \cdot \ldots \cdot \lambda s_{q-1} \cdot [\![v]\!]_\oplus(s_0, \ldots, s_{\ell-1}, s_\ell \oplus s_{\ell+1}, s_{\ell+2}, \ldots, s_{q-1})$ (since $q \geq \ell+2$, by (η)) $= \lambda s_n \cdot \ldots \cdot \lambda s_{q-1} \cdot [\![u]\!]_\oplus(s_0, \ldots, s_{q-1})$ $= [\![u]\!]_\oplus(s_0, \ldots, s_{n-1})$;
- if $q > m > n$, then $\lambda x_n \cdot \ldots \cdot \lambda x_{m-1} \cdot [\![u]\!]_\oplus(s_0, \ldots, s_{n-1}, x_n, \ldots, x_{m-1}) = \lambda x_n \cdot \ldots \cdot \lambda x_{m-1} \cdot \lambda x_m \cdot \ldots \cdot \lambda x_{q-1} \cdot [\![u]\!]_\oplus(s_0, \ldots, s_{n-1}, x_n, \ldots, x_{q-1}) = \lambda s_n \cdot \ldots \cdot \lambda s_{m-1} \cdot \lambda s_m \cdot \ldots \cdot \lambda s_{q-1} \cdot [\![u]\!]_\oplus(s_0, \ldots, s_{q-1})$ (by α-renaming) $= [\![u]\!]_\oplus(s_0, \ldots, s_{n-1})$.

Finally, if u is of the form $S_\ell(v, w)$, then let $q =_{\rm df} \dim u = \max(\ell, \dim v - 1)$; we have three cases:

- If $m > n \geq q$, then $\lambda x_n \cdot \ldots \cdot \lambda x_{m-1} \cdot [\![u]\!]_\oplus(s_0, \ldots, s_{n-1}, x_n, \ldots, x_{m-1}) = \lambda x_n \cdot \ldots \cdot \lambda x_{m-1} \cdot [\![v]\!]_\oplus(s_0, \ldots, s_{\ell-1}, [\![w]\!]_\oplus(s_0, \ldots, s_{\ell-1}), s_\ell, \ldots, s_{n-1}, x_n, \ldots, x_{m-1})$ $\longrightarrow^* [\![v]\!]_\oplus(s_0, \ldots, s_{\ell-1}, [\![w]\!]_\oplus(s_0, \ldots, s_{\ell-1}), s_\ell, \ldots, s_{n-1})$ (by induction) $= [\![u]\!]_\oplus(s_0, \ldots, s_{n-1})$;
- if $m \geq q > n$, then we have $\lambda x_n \cdot \ldots \cdot \lambda x_{m-1} \cdot [\![u]\!]_\oplus(s_0, \ldots, s_{n-1}, x_n, \ldots, x_{m-1}) = \lambda s_n \cdot \ldots \cdot \lambda s_{m-1} \cdot [\![u]\!]_\oplus(s_0, \ldots, s_{m-1})$ (by α-renaming) $= \lambda s_n \cdot \ldots \cdot \lambda s_{m-1} \cdot [\![v]\!]_\oplus(s_0, \ldots, s_{\ell-1}, [\![w]\!]_\oplus(s_0, \ldots, s_{\ell-1}), s_\ell, \ldots, s_{m-1}) \longrightarrow^* \lambda s_n \cdot \ldots \cdot \lambda s_{q-1} \cdot [\![v]\!]_\oplus(s_0, \ldots, s_{\ell-1}, [\![w]\!]_\oplus(s_0, \ldots, s_{\ell-1}), s_\ell, \ldots, s_{q-1})$ (by induction) $= [\![u]\!]_\oplus(s_0, \ldots, s_{n-1})$;
- if $q > m > n$, then $\lambda x_n \cdot \ldots \cdot \lambda x_{m-1} \cdot [\![u]\!]_\oplus(s_0, \ldots, s_{n-1}, x_n, \ldots, x_{m-1}) = \lambda x_n \cdot \ldots \cdot \lambda x_{m-1} \cdot \lambda x_m \cdot \ldots \cdot \lambda x_{q-1} \cdot [\![u]\!]_\oplus(s_0, \ldots, s_{n-1}, x_n, \ldots, x_{q-1}) = [\![u]\!]_\oplus(s_0, \ldots, s_{n-1})$.

Now, we claim that: (e) for every rule $l \to r$ in SKInT$_\eta$, for every $\lambda_{\oplus\epsilon}$-terms $s_0, \ldots, s_{n-1}$, $[\![l]\!]_\oplus(s_0, \ldots, s_{n-1}) \longrightarrow^* [\![r]\!]_\oplus(s_0, \ldots, s_{n-1})$ (resp. $\longrightarrow^+$ in the case of rules (SI_ℓ), (SK_ℓ), $(S_\ell I_{\mathcal{L}})$ and (ηS_ℓ)) whenever the left-hand side makes sense and $n \geq \dim l$. Indeed, examine each rule in turn:

- (SI_ℓ): $l = S_\ell(I_\ell, w)$, $r = w$. Since $n \geq \dim l = \ell$:

$$\begin{array}{ll}
[\![l]\!]_\oplus(s_0, \ldots, s_{n-1}) & \\
= [\![I_\ell]\!]_\oplus(s_0, \ldots, s_{\ell-1}, [\![w]\!]_\oplus(s_0, \ldots, s_{\ell-1}), s_\ell, \ldots, s_{n-1}) & \\
= \epsilon(s_0 \oplus \ldots \oplus s_{\ell-1} \oplus [\![w]\!]_\oplus(s_0, \ldots, s_{\ell-1}))s_\ell \ldots s_{n-1} & \\
\longrightarrow^* \epsilon([\![w]\!]_\oplus(s_0, \ldots, s_{\ell-1}))s_\ell \ldots s_{n-1} & \text{(by } (\oplus-)\text{)} \\
\longrightarrow ([\![w]\!]_\oplus(s_0, \ldots, s_{\ell-1}))s_\ell \ldots s_{n-1} & \text{(by } (\epsilon)\text{)} \\
\longrightarrow^* [\![w]\!]_\oplus(s_0, \ldots, s_{n-1}) & \text{(by (c))} \\
= [\![r]\!]_\oplus(s_0, \ldots, s_{n-1}) &
\end{array}$$

- (SK_ℓ): $l = S_\ell(K_\ell(u), v)$, $r = u$. Since $n \geq \dim l = \max(\ell+1, \dim u)$:

$$\begin{array}{ll}
[\![l]\!]_\oplus(s_0, \ldots, s_{n-1}) & \\
= [\![K_\ell(u)]\!]_\oplus(s_0, \ldots, s_{\ell-1}, [\![w]\!]_\oplus(s_0, \ldots, s_{\ell-1}), s_\ell, \ldots, s_{n-1}) & \text{(since } n \geq \ell\text{)} \\
= [\![u]\!]_\oplus(s_0, \ldots, s_{\ell-1}, [\![w]\!]_\oplus(s_0, \ldots, s_{\ell-1}) \oplus s_\ell, s_{\ell+1}, \ldots, s_{n-1}) & \text{(since } n \geq \ell+1\text{)} \\
\longrightarrow^+ [\![u]\!]_\oplus(s_0, \ldots, s_{\ell-1}, s_\ell, s_{\ell+1}, \ldots, s_{n-1}) & \text{(by } (\oplus-) \text{ and (a))} \\
= [\![r]\!]_\oplus(s_0, \ldots, s_{n-1}) &
\end{array}$$

- $(S_\ell I_{\mathcal{L}})$: $l = S_\ell(I_{\mathcal{L}}, w)$, $0 \leq \ell < \mathcal{L}$, and $r = I_{\mathcal{L}-1}$. Then $n \geq \dim l = \mathcal{L}$ and:

$$\begin{array}{ll}
[\![l]\!]_\oplus(s_0, \ldots, s_{n-1}) & \\
= [\![I_{\mathcal{L}}]\!]_\oplus(s_0, \ldots, s_{\ell-1}, [\![w]\!]_\oplus(s_0, \ldots, s_{\ell-1}), s_\ell, \ldots, s_{n-1}) & \text{(since } n \geq \ell\text{)} \\
= \epsilon(s_0 \oplus \ldots \oplus s_{\ell-1} \oplus [\![w]\!]_\oplus(s_0, \ldots, s_{\ell-1}) \oplus s_\ell \oplus \ldots \oplus s_{\mathcal{L}})s_{\mathcal{L}+1} \ldots s_{n-1} & \\
\longrightarrow \epsilon(s_0 \oplus \ldots \oplus s_{\ell-1} \oplus s_\ell \oplus \ldots \oplus s_{\mathcal{L}})s_{\mathcal{L}+1} \ldots s_{n-1} & \text{(by } (\oplus-)\text{, since } \mathcal{L} > \ell\text{)} \\
= [\![r]\!]_\oplus(s_0, \ldots, s_{n-1}) &
\end{array}$$

- $(S_\ell K_\mathcal{L})$: $l = S_\ell(K_\mathcal{L}(u), w)$, $r = K_{\mathcal{L}-1}(S_\ell(u, w))$. We have $n \geq \dim l = \max(\mathcal{L}+1, \dim u)$, and:

$$\begin{aligned}
&[\![l]\!]_\oplus(s_0, \ldots, s_{n-1}) \\
&= [\![K_\mathcal{L}(u)]\!]_\oplus(s_0, \ldots, s_{\ell-1}, [\![w]\!]_\oplus(s_0, \ldots, s_{\ell-1}), s_\ell, \ldots, s_{n-1}) && \text{(since } n \geq \ell\text{)} \\
&= [\![u]\!]_\oplus(s_0, \ldots, s_{\ell-1}, [\![w]\!]_\oplus(s_0, \ldots, s_{\ell-1}), s_\ell, \ldots, s_{\mathcal{L}-2}, s_{\mathcal{L}-1} \oplus s_\mathcal{L}, s_{\mathcal{L}+1}, \ldots, s_{n-1}) && \text{(since } n \geq \mathcal{L}+1\text{,} \\
& && \text{and observing that } \mathcal{L}-1 \geq \ell\text{)} \\
&= [\![S_\ell(u, w)]\!]_\oplus(s_0, \ldots, s_\mathcal{L}, s_{\mathcal{L}-1} \oplus s_\mathcal{L}, s_{\mathcal{L}+1}, \ldots, s_{n-1}) \\
&= [\![r]\!]_\oplus(s_0, \ldots, s_{n-1})
\end{aligned}$$

- $(S_\ell S_\mathcal{L})$: $l = S_\ell(S_\mathcal{L}(u, v), w)$, $r = S_{\mathcal{L}-1}(S_\ell(u, w), S_\ell(v, w))$. Since $n \geq \dim l = \max(\mathcal{L}-1, \dim u - 2)$:

$$\begin{aligned}
&[\![l]\!]_\oplus(s_0, \ldots, s_{n-1}) \\
&= [\![S_\mathcal{L}(u, v)]\!]_\oplus(s_0, \ldots, s_{\ell-1}, [\![w]\!]_\oplus(s_0, \ldots, s_{\ell-1}), s_\ell, \ldots, s_{n-1}) && \text{(since } n \geq \ell\text{)} \\
&= [\![u]\!]_\oplus(s_0, \ldots, s_{\ell-1}, [\![w]\!]_\oplus(s_0, \ldots, s_{\ell-1}), s_\ell, \ldots, s_{\mathcal{L}-2}, \\
&\qquad [\![v]\!]_\oplus(s_0, \ldots, s_{\ell-1}, [\![w]\!]_\oplus(s_0, \ldots, s_{\ell-1}), s_\ell, \ldots, s_{\mathcal{L}-2}), \\
&\qquad s_{\mathcal{L}-1}, \ldots, s_{n-1}) && \text{(since } n \geq \mathcal{L}-1\text{)} \\
&= [\![S_\ell(u, w)]\!]_\oplus(s_0, \ldots, s_{\mathcal{L}-2}, \\
&\qquad [\![S_\ell(v, w)]\!]_\oplus(s_0, \ldots, s_{\mathcal{L}-2}), \\
&\qquad s_{\mathcal{L}-1}, \ldots, s_{n-1}) \\
&= [\![r]\!]_\oplus(s_0, \ldots, s_{n-1})
\end{aligned}$$

- $(K_\ell I_\mathcal{L})$: $l = K_\ell(I_{\mathcal{L}-1})$, $r = I_\mathcal{L}$. We have $n \geq \dim l = \mathcal{L}+1$, and:

$$\begin{aligned}
&[\![l]\!]_\oplus(s_0, \ldots, s_{n-1}) \\
&= [\![I_{\mathcal{L}-1}]\!]_\oplus(s_0, \ldots, s_{\ell-1}, s_\ell \oplus s_{\ell+1}, s_{\ell+2}, \ldots, s_{n-1}) && \text{(since } n \geq \ell+2\text{)}
\end{aligned}$$

If $\mathcal{L} = \ell + 1$, then this equals:

$$\begin{aligned}
&[\![I_\ell]\!]_\oplus(s_0, \ldots, s_{\ell-1}, s_\ell \oplus s_{\ell+1}, s_{\ell+2}, \ldots, s_{n-1}) \\
&= \epsilon(s_0 \oplus \ldots \oplus s_{\ell-1} \oplus (s_\ell \oplus s_{\ell+1}))s_{\ell+2} \ldots s_{n-1} \\
&= \epsilon(s_0 \oplus \ldots \oplus s_{\ell-1} \oplus s_\ell \oplus s_{\ell+1})s_{\ell+2} \ldots s_{n-1} && \text{(since by convention } \oplus \text{ is right-associative)} \\
&= [\![r]\!]_\oplus(s_0, \ldots, s_{n-1})
\end{aligned}$$

If $\mathcal{L} > \ell + 1$, then instead the interpretation of the left-hand side is:

$$\begin{aligned}
&\epsilon(s_0 \oplus \ldots \oplus s_{\ell-1} \oplus (s_\ell \oplus s_{\ell+1}) \oplus s_{\ell+2} \oplus \ldots \oplus s_\mathcal{L})s_{\mathcal{L}+1} \ldots s_{n-1} \\
&\longrightarrow \epsilon(s_0 \oplus \ldots \oplus s_{\ell-1} \oplus s_\ell \oplus s_{\ell+1} \oplus s_{\ell+2} \oplus \ldots \oplus s_\mathcal{L})s_{\mathcal{L}+1} \ldots s_{n-1} && \text{(by } (\oplus)\text{)} \\
&= [\![r]\!]_\oplus(s_0, \ldots, s_{n-1})
\end{aligned}$$

- $(K_\ell K_\mathcal{L})$: $l = K_\ell(K_{\mathcal{L}-1}(u))$, $r = K_\mathcal{L}(K_\ell(u))$. Then $n \geq \dim l = \max(\mathcal{L}+2, \dim u + 2)$, and:

$$\begin{aligned}
&[\![l]\!]_\oplus(s_0, \ldots, s_{n-1}) \\
&= [\![K_{\mathcal{L}-1}(u)]\!]_\oplus(s_0, \ldots, s_{\ell-1}, s_\ell \oplus s_{\ell+1}, s_{\ell+2}, \ldots, s_{n-1}) && \text{(since } n \geq \ell+2\text{)}
\end{aligned}$$

Now, if $\mathcal{L} = \ell + 1$, this equals:

$$\begin{aligned}
&[\![K_\ell(u)]\!]_\oplus(s_0, \ldots, s_{\ell-1}, s_\ell \oplus s_{\ell+1}, s_{\ell+2}, \ldots, s_{n-1}) \\
&= [\![u]\!]_\oplus(s_0, \ldots, s_{\ell-1}, (s_\ell \oplus s_{\ell+1}) \oplus s_{\ell+2}, s_{\ell+3}, \ldots, s_{n-1}) \\
&\longrightarrow^+ [\![u]\!]_\oplus(s_0, \ldots, s_{\ell-1}, s_\ell \oplus (s_{\ell+1} \oplus s_{\ell+2}), s_{\ell+3}, \ldots, s_{n-1}) && \text{(by } (\oplus) \text{ and (a))} \\
&= [\![K_\ell(u)]\!]_\oplus(s_0, \ldots, s_{\ell-1}, s_\ell, s_{\ell+1} \oplus s_{\ell+2}, s_{\ell+3}, \ldots, s_{n-1}) \\
&= [\![r]\!]_\oplus(s_0, \ldots, s_{n-1})
\end{aligned}$$

And if $\mathcal{L} > \ell + 1$, then the left-hand side is:

$$\begin{aligned}
&[\![u]\!]_\oplus(s_0, \ldots, s_{\ell-1}, s_\ell \oplus s_{\ell+1}, s_{\ell+2}, \ldots, s_{\mathcal{L}-1}, s_\mathcal{L} \oplus s_{\mathcal{L}+1}, s_{\mathcal{L}+2}, \ldots, s_{n-1}) && \text{(since } n \geq \mathcal{L}+2\text{)} \\
&= [\![K_\ell(u)]\!]_\oplus(s_0, \ldots, s_{\mathcal{L}-1}, s_\mathcal{L} \oplus s_{\mathcal{L}+1}, s_{\mathcal{L}+2}, \ldots, s_{n-1}) \\
&= [\![r]\!]_\oplus(s_0, \ldots, s_{n-1})
\end{aligned}$$

- $(K_\ell S_{\mathcal{L}+1})$: $l = K_\ell(S_{\mathcal{L}}(u,v))$, $r = S_{\mathcal{L}+1}(K_\ell(u), K_\ell(v))$. Then $n \geq \dim l = \max(\mathcal{L}+1, \dim u$, and:

$$\begin{array}{ll}
[\![l]\!]_\oplus(s_0,\ldots,s_{n-1}) & \\
= [\![S_{\mathcal{L}}(u,v)]\!]_\oplus(s_0,\ldots,s_{\ell-1}, s_\ell \oplus s_{\ell+1}, s_{\ell+2},\ldots,s_{n-1}) & \text{(since } n \geq \ell+2\text{)} \\
= [\![u]\!]_\oplus(s_0,\ldots,s_{\ell-1}, s_\ell \oplus s_{\ell+1}, s_{\ell+2},\ldots,s_{\mathcal{L}}, & \\
\qquad [\![v]\!]_\oplus(s_0,\ldots,s_{\ell-1}, s_\ell \oplus s_{\ell+1}, s_{\ell+2},\ldots,s_{\mathcal{L}}), & \\
\qquad s_{\mathcal{L}+1},\ldots,s_{n-1}) & \text{(since } n \geq \mathcal{L}+1\text{)} \\
= [\![K_\ell(u)]\!]_\oplus(s_0,\ldots,s_{\mathcal{L}}, [\![K_\ell(v)]\!]_\oplus(s_0,\ldots,s_{\mathcal{L}}), s_{\mathcal{L}+1},\ldots,s_{n-1}) & \\
= [\![r]\!]_\oplus(s_0,\ldots,s_{n-1}) &
\end{array}$$

- (ηS_ℓ): $l = S_{\ell+1}(K_\ell(u), I_\ell)$, $r = u$. Since $n \geq \dim l = \max(\ell+1, \dim u)$:

$$\begin{array}{ll}
[\![l]\!]_\oplus(s_0,\ldots,s_{n-1}) & \\
= [\![K_\ell(u)]\!]_\oplus(s_0,\ldots,s_\ell, [\![I_\ell]\!]_\oplus(s_0,\ldots,s_\ell), s_{\ell+1},\ldots,s_{n-1}) & \text{(since } n \geq \ell+1\text{)} \\
= [\![K_\ell(u)]\!]_\oplus(s_0,\ldots,s_\ell, \epsilon(s_0 \oplus \ldots \oplus s_\ell), s_{\ell+1},\ldots,s_{n-1}) & \\
\longrightarrow^+ [\![K_\ell(u)]\!]_\oplus(s_0,\ldots,s_\ell, s_\ell, s_{\ell+1},\ldots,s_{n-1}) & \text{(by } (\epsilon)\text{, } (\oplus-)\text{ and (a))} \\
= [\![u]\!]_\oplus(s_0,\ldots,s_{\ell-1}, s_\ell \oplus s_\ell, s_{\ell+1},\ldots,s_{n-1}) & \\
\longrightarrow^+ [\![u]\!]_\oplus(s_0,\ldots,s_{\ell-1}, s_\ell, s_{\ell+1},\ldots,s_{n-1}) & \text{(by } (\oplus-)\text{ and (a))} \\
= [\![r]\!]_\oplus(s_0,\ldots,s_{n-1}) &
\end{array}$$

It follows that: (f) for every rule $l \to r$ in SKInT_η, for every $\lambda_{\oplus\epsilon}$-terms $s_0, \ldots, s_{n-1}$, $[\![l]\!]_\oplus(s_0,\ldots,s_{n-1}) \longrightarrow^* [\![r]\!]_\oplus(s_0,\ldots,s_{n-1})$ (resp. $\longrightarrow^+$ in the case of rules (SI_ℓ), (SK_ℓ), $(S_\ell I_{\mathcal{L}})$ and (ηS_ℓ)) whenever the left-hand side makes sense. Indeed, if $n \geq \dim l$, this is exactly (e). Otherwise, namely if $n < q =_{\mathrm{df}} \dim l$, then $[\![l]\!]_\oplus(s_0,\ldots,s_{n-1}) = \lambda x_n \cdot \ldots \cdot \lambda x_{q-1} \cdot [\![l]\!]_\oplus(s_0,\ldots,s_{n-1},x_n,\ldots,x_{q-1}) \longrightarrow^* \lambda x_n \cdot \ldots \cdot \lambda x_{q-1} \cdot [\![r]\!]_\oplus(s_0,\ldots,s_{n-1},x_n,\ldots,x_{q-1})$ (resp. $\longrightarrow^+$; by (e)) $\longrightarrow^* [\![r]\!]_\oplus(s_0,\ldots,s_{n-1})$ (by (d)).

It also follows that: (g) for every reduction step $u \longrightarrow v$ in SKInT_η, for every $\lambda_{\oplus\epsilon}$-terms $s_0, \ldots, s_{n-1}$, $[\![u]\!]_\oplus(s_0,\ldots,s_{n-1}) \longrightarrow^* [\![v]\!]_\oplus(s_0,\ldots,s_{n-1})$ (resp. $\longrightarrow^+$ in the case of rules (SI_ℓ), (SK_ℓ), $(S_\ell I_{\mathcal{L}})$ and (ηS_ℓ)) whenever the left-hand side makes sense. Indeed, write u as $\mathcal{C}[l]$, where $\mathcal{C}$ is a context, $l \to r$ is some rule, and $v = \mathcal{C}[r]$. Contexts are expressions generated by the grammar:

$$\mathcal{C} ::= [] \mid K_\ell(\mathcal{C}) \mid S_\ell(\mathcal{C}, w) \mid S_\ell(w, \mathcal{C})$$

where w is a term, $\ell \geq 0$, and $[]$ is the empty context. $\mathcal{C}[u]$ denotes $\mathcal{C}$ with the hole $[]$ replaced by u, whether u is a term or another context.

We show (g) by structural induction on the context $\mathcal{C}$, first assuming $n \geq \dim u$ (if $n < \dim u$, then the same argument applies as when we deduced (f) from (e) in each of the following cases). If $\mathcal{C}$ is the empty context, then this is (f). Otherwise, if $\mathcal{C}$ is of the form $K_\ell(\mathcal{C}_1)$, then $[\![u]\!]_\oplus(s_0,\ldots,s_{n-1}) = [\![\mathcal{C}_1[l]]\!]_\oplus(s_0,\ldots,s_{\ell-1}, s_\ell \oplus s_{\ell+1}, s_{\ell+2},\ldots,s_{n-1}) \longrightarrow^* [\![\mathcal{C}_1[r]]\!]_\oplus(s_0,\ldots,s_{\ell-1}, s_\ell \oplus s_{\ell+1}, s_{\ell+2},\ldots,s_{n-1})$ (resp. $\longrightarrow^+$; by induction hypothesis) $= [\![v]\!]_\oplus(s_0,\ldots,s_{n-1})$. If $\mathcal{C}$ is of the form $S_\ell(\mathcal{C}_1, w)$, then we see that $[\![u]\!]_\oplus(s_0,\ldots,s_{n-1}) = [\![\mathcal{C}_1[l]]\!]_\oplus(s_0,\ldots,s_{\ell-1}, [\![w]\!]_\oplus(s_0,\ldots,s_{\ell-1}), s_\ell,\ldots,s_{n-1}) \longrightarrow^* [\![\mathcal{C}_1[r]]\!]_\oplus(s_0,\ldots,s_{\ell-1}, [\![w]\!]_\oplus(s_0,\ldots,s_{\ell-1}), s_\ell,\ldots,s_{n-1})$
(resp. $\longrightarrow^+$; by induction hypothesis) $= [\![v]\!]_\oplus(s_0,\ldots,s_{n-1})$. And if $\mathcal{C}$ is of the form $S_\ell(w, \mathcal{C}_1)$, then we observe that $[\![u]\!]_\oplus(s_0,\ldots,s_{n-1}) = [\![w]\!]_\oplus(s_0,\ldots,s_{\ell-1}, [\![\mathcal{C}_1[l]]\!]_\oplus(s_0,\ldots,s_{\ell-1}), s_\ell,\ldots,s_{n-1}) \longrightarrow^* [\![w]\!]_\oplus(s_0,\ldots,s_{\ell-1}, [\![\mathcal{C}_1[r]]\!]_\oplus(s_0,\ldots,s_{\ell-1}), s_\ell,\ldots,s_{n-1})$ (resp. $\longrightarrow^+$; by induction hypothesis and (a)) $= [\![v]\!]_\oplus(s_0,\ldots,s_{n-1})$.

We also claim that: (h) If $u \longrightarrow^s v$ in SKInT_η (notice that this is a spine-reduction), then $[\![u]\!]_\oplus(s_0,\ldots,s_{n-1}) \longrightarrow^* [\![v]\!]_\oplus(s_0,\ldots,s_{n-1})$ (resp. $\longrightarrow^+$ if $u \longrightarrow^s v$ by (SI_ℓ), (SK_ℓ), $(S_\ell I_{\mathcal{L}})$, or (ηS_ℓ)) by head-reductions in $\lambda_{\oplus\epsilon}$. We invite the reader to check this on all the steps (a)–(g) above.

H $S\omega$ and expansions

We check that $S\omega$-typings are preserved by using all rules except (SI_ℓ), (SK_ℓ) and (ηS_ℓ) as expansions instead of contractions:

- $(S_\ell I_\mathcal{L})$: $l = S_\ell(I_\mathcal{L}, w)$, $r = I_{\mathcal{L}-1}$; then any type τ of r is of the form $\mu_0 \to \ldots \to \mu_{\mathcal{L}-2} \to \mu \wedge \tau' \to \tau'$, and l also has this type.

- $(S_\ell S_\mathcal{L})$: $l = S_\ell(S_\mathcal{L}(u, v), w)$, $r = S_{\mathcal{L}-1}(S_\ell(u, w), S_\ell(v, w))$. Any type τ that r may have is of the form $\mu_0 \to \ldots \to \mu_{\mathcal{L}-1} \to \tau'$, and is obtained from a typing derivation that gave u a type of the form:

$$\mu_0 \to \ldots \to \mu_{\ell-1} \to \mu \to \mu_\ell \to \ldots \to \mu_{\mathcal{L}-1} \to \mu' \to \tau'$$

  where μ is of the form $[\tau_1, \ldots, \tau_k]$, μ' is of the form $[\tau'_1, \ldots, \tau'_{k'}]$, w has been assigned all types of the form $\mu_0 \to \ldots \to \mu_{\ell-1} \to \tau_i$, $1 \leq i \leq k$, and v has been assigned all types of the form $\mu_0 \to \ldots \to \mu_{\ell-1} \to \mu \to \mu_\ell \to \ldots \to \mu_{\mathcal{L}-1} \to \tau'_{i'}$, $1 \leq i' \leq k'$. Then l can be given the type τ as well.

- $(S_\ell K_\mathcal{L})$: $l = S_\ell(K_\mathcal{L}(u), w)$, $r = K_{\mathcal{L}-1}(S_\ell(u, w))$. Any type τ of r is of the form $\mu_0 \to \ldots \to \mu_{\mathcal{L}-1} \to \mu_\mathcal{L} \to \tau'$, and is obtained from a typing derivation that gave u a type of the form:

$$\mu_0 \to \ldots \to \mu_{\ell-1} \to \mu \to \mu_\ell \to \ldots \to \mu_{\mathcal{L}-2} \to \mu_\mathcal{L} \to \tau'$$

  where μ is of the form $[\tau_1, \ldots, \tau_k]$, and w has been assigned all types $\mu_0 \to \ldots \to \mu_{\ell-1} \to \tau_i$, $1 \leq i \leq k$. It follows that l can be given the same type as r.

- $(K_\ell I_\mathcal{L})$: $l = K_\ell(I_{\mathcal{L}-1})$, $r = I_\mathcal{L}$. Any type τ of r is of the form $\mu_0 \to \ldots \to \mu_{\mathcal{L}-1} \to \mu \wedge \tau' \to \tau'$, and this is also a type for l.

- $(K_\ell S_{\mathcal{L}+1})$: $l = K_\ell(S_\mathcal{L}(u, v))$, $r = S_{\mathcal{L}+1}(K_\ell(u), K_\ell(v))$. Any type τ of r is of the form $\mu_0 \to \ldots \to \mu_\mathcal{L} \to \tau'$, and is obtained by giving u a type of the form:

$$\mu_0 \to \ldots \to \mu_{\ell-1} \to \mu_{\ell+1} \to \ldots \to \mu_\mathcal{L} \to \mu \to \tau'$$

  where μ is of the form $[\tau_1, \ldots, \tau_k]$, and w has been assigned all types $\mu_0 \to \ldots \to \mu_{\ell-1} \to \mu_{\ell+1} \to \ldots \to \mu_\mathcal{L} \to \tau_i$, $1 \leq i \leq k$. It follows that l can be given the same type as r.

- $(K_\ell K_\mathcal{L})$: $l = K_\ell(K_{\mathcal{L}-1}(u))$, $r = K_\mathcal{L}(K_\ell(u))$. Any type τ of r is of the form $\mu_0 \to \ldots \to \mu_\mathcal{L} \to \mu_{\mathcal{L}+1} \to \tau'$, and is obtained by giving u a type of the form:

$$\mu_0 \to \ldots \to \mu_{\ell-1} \to \mu_{\ell+1} \to \ldots \to \mu_{\mathcal{L}-1} \to \mu_{\mathcal{L}+1} \to \tau'$$

  Then l can be given the same type as r.

I Quasi-commutation of η over the other rules

We show that: (a) if:

$$u \longrightarrow^\eta v \longrightarrow^{\text{SKInT}} w$$

then:

$$u \longrightarrow^{\text{SKInT}^+} v' \longrightarrow^{\eta^*} w$$

A *context* $\mathcal{C}$ is a term with one hole, written $[]$, i.e.:

$$\mathcal{C} ::= [] \mid S_\ell(\mathcal{C}, u) \mid S_\ell(u, \mathcal{C}) \mid K_\ell(\mathcal{C})$$

where u ranges over SKInT-terms and $\ell \geq 0$. $\mathcal{C}[v]$ denotes $\mathcal{C}$ where the hole has been replaced by the term v; similarly, $\mathcal{C}[\mathcal{C}']$ denotes a context obtained by plugging the context $\mathcal{C}'$ into the hole of $\mathcal{C}$.

Consider a given rewrite:

$$u \longrightarrow^\eta v \longrightarrow^{\text{SKInT}} w$$

Then u is of the form $\mathcal{C}[S_{\ell+1}(K_\ell(t), I_\ell)]$, $v = \mathcal{C}[t]$ and also $v = \mathcal{C}'[l\sigma]$, $w = \mathcal{C}'[r\sigma]$ for some instance of a rule $l \to r$ in SKInT by some substitution σ.

If the SKInT and η-redexes are side-by-side, namely, neither $\mathcal{C}$ nor $\mathcal{C}'$ is a subcontext of the other, then we have:

$$u \longrightarrow^{\text{SKInT}} v' \longrightarrow^{\eta} w$$

for some term v'.

If the η-redex t contains the ΣT-redex, namely if $\mathcal{C}$ is included in $\mathcal{C}'$ (or possibly equal to it), then again:

$$u \longrightarrow^{\text{SKInT}} v' \longrightarrow^{\eta} w$$

for some term v'. In fact, if $\mathcal{C}' = \mathcal{C}[\mathcal{C}_1]$, then $v' = \mathcal{C}[S_{\ell+1}(K_\ell(\mathcal{C}_1[r\sigma]), I_\ell)]$.

If the SKInT-redex is above the η-redex, namely if $\mathcal{C}'$ is included in $\mathcal{C}$, i.e. $\mathcal{C} = \mathcal{C}'[\mathcal{C}_1]$ for some context $\mathcal{C}_1$, then $v = \mathcal{C}[t] = \mathcal{C}'[\mathcal{C}_1[t]]$ and on the other hand $v = \mathcal{C}'[l\sigma]$, so that $l\sigma = \mathcal{C}_1[t]$. We then have two cases: either the distinguished occurrence of t is at or below some variable position in l, and then there is another substitution σ' such that $l\sigma' = \mathcal{C}_1[S_{\ell+1}(K_\ell(t), I_\ell)]$, so that:

$$u \longrightarrow^{\text{SKInT}} v' \longrightarrow^{\eta^*} w$$

where $v' = \mathcal{C}'[r\sigma']$ and w is obtained by contracting the residuals of the occurrence of $S_{\ell+1}(K_\ell(t), I_\ell)$ in v' by η. Or $l\sigma = \mathcal{C}_1[t]$ but there is an overlap between the left-hand side l and the term t. (This is a kind of critical pair, between η and the inverse relation SKInT^{-1}.) We have eight critical pairs to consider, each corresponding to a rule in SKInT:

1. $u = S_i(S_{\ell+1}(K_\ell(I_j), I_\ell), s) \longrightarrow^{(\eta S_\ell)} v = S_i(I_j, s) \longrightarrow^{(S_i I_j)} I_{j-1} = w$, with $0 \leq i < j$.
   (a) If $i < \ell$, then we have $u \longrightarrow^{(S_i S_{\ell+1})} S_\ell(S_i(K_\ell(I_j), s), S_i(I_\ell, s)) \longrightarrow^{(S_i K_\ell),(S_i I_\ell)} S_\ell(K_{\ell-1}(S_i(I_j, s)), I_{\ell-1}) \longrightarrow^{(S_i I_j)} S_\ell(K_{\ell-1}(I_{j-1}), I_{\ell-1}) \longrightarrow^{(\eta S_{\ell-1})} I_{j-1} = w$.
   (b) If $i \geq \ell$, then $u \longrightarrow^{(K_\ell I_{j1})} S_i(S_{\ell+1}(I_{j+1}, I_\ell), s)$ (because $\ell \leq i < j$) $\longrightarrow^{(S_{\ell+1} I_{j+1})} S_i(I_j, s) \longrightarrow^{(S_i I_j)} I_{j-1} = w$.

2. $u = S_i(S_{\ell+1}(K_\ell(K_j(s)), I_\ell), t) \longrightarrow^{(\eta S_\ell)} v = S_i(K_j(s), t) \longrightarrow^{(S_i K_j)} K_{j-1}(S_i(s,t)) = w$, with $0 \leq i < j$.
   (a) If $i < \ell$, then we have $u \longrightarrow^{(S_i S_{\ell+1})} S_\ell(S_i(K_\ell(K_j(s)), t), S_i(I_\ell, t)) \longrightarrow^{(S_i K_\ell),(S_i K_j),(S_i I_\ell)} S_\ell(K_{\ell-1}(K_{j-1}(S_i(s,t))), I_{\ell-1}) \longrightarrow^{(\eta S_{\ell-1})} K_{j-1}(S_i(s,t)) = w$.
   (b) If $i \geq \ell$, then $u \longrightarrow^{(K_\ell K_{j+1})} S_i(S_{\ell+1}(K_{j+1}(K_\ell(s)), I_\ell), t)$ (because $\ell \leq i \leq j$) $\longrightarrow^{(S_{\ell+1} K_{j+1})} S_i(K_j(S_{\ell+1}(K_\ell(s), I_\ell)), t)$ (because $\ell \leq i < j$) $\longrightarrow^{(S_i K_j)} K_{j-1}(S_i(S_{\ell+1}(K_\ell(s), I_\ell), t)) \longrightarrow^{(\eta S_\ell)} K_{j-1}(S_i(s,t)) = w$.

3. $u = S_i(S_{\ell+1}(K_\ell(S_j(s,t)), I_\ell), t') \longrightarrow^{(\eta S_\ell)} v = S_i(S_j(s,t), t') \longrightarrow^{(S_i S_j)} S_{j-1}(S_i(s,t'), S_i(t,t')) = w$, with $0 \leq i < j$.
   (a) If $i < \ell$, then we have $u \longrightarrow^{(S_i S_{\ell+1})} S_\ell(S_i(K_\ell(S_j(s,t)), t'), S_i(I_\ell, t')) \longrightarrow^{(S_i K_\ell),(S_i S_j),(S_i I_\ell)} S_\ell(K_{\ell-1}(S_{j-1}(S_i(s,t'), S_i(t,t'))), I_{\ell-1}) \longrightarrow^{(\eta S_{\ell-1})} S_{j-1}(S_i(s,t'), S_i(t,t')) = w$.
   (b) If $i \geq \ell$, then we have $u \longrightarrow^{(K_\ell S_j)} S_i(S_{\ell+1}(S_{j+1}(K_\ell(s), K_\ell(t)), I_\ell), t')$ (this rule is indeed applicable since $\ell \leq i < j$) $\longrightarrow^{(S_{\ell+1} S_{j+1})} S_i(S_j(S_{\ell+1}(K_\ell(s), I_\ell), S_{\ell+1}(K_\ell(t), I_\ell)), t') \longrightarrow^{(S_i S_j)} S_{j-1}(S_i(S_{\ell+1}(K_\ell(s), I_\ell), t'), S_i(S_{\ell+1}(K_\ell(t), I_\ell), t')) \longrightarrow^{(\eta S_\ell)} S_{j-1}(S_i(s,t'), S_i(S_{\ell+1}(K_\ell(t), I_\ell), t')) \longrightarrow^{(\eta S_\ell)} S_{j-1}(S_i(s,t'), S_i(t,t')) = w$.

4. $u = K_i(S_{\ell+1}(K_\ell(I_{j-1}), I_\ell)) \longrightarrow^{(\eta S_\ell)} v = K_i(I_{j-1}) \longrightarrow^{(K_i I_j)} I_j = w$, with $0 \leq i < j$.
   (a) If $i \leq \ell$, then we have $u \longrightarrow (K_i S_{\ell+1}) S_{\ell+2}(K_i(K_\ell(I_{j-1})), K_i(I_\ell)) \longrightarrow^{(K_i K_{\ell+1}),(K_i I_j),(K_i I_{\ell+1})} S_{\ell+2}(K_{\ell+1}(I_j), I_{\ell+1}) \longrightarrow^{(\eta S_{\ell+1})} I_j = w$.
   (b) If $i > \ell$, then $u \longrightarrow^{(K_\ell I_j)} K_i(S_{\ell+1}(I_j, I_\ell))$ (because $\ell \leq i < j$) $\longrightarrow^{(S_{\ell+1} I_j)} K_i(I_{j-1})$ (because $\ell < i < j$, so that $\ell + 1 < j$) $\longrightarrow^{(K_i I_j)} I_j = w$.

5. $u = K_i(S_{\ell+1}(K_\ell(K_{j-1}(s)), I_\ell)) \longrightarrow^{(\eta S_\ell)} v = K_i(K_{j-1}(s)) \longrightarrow^{(K_i K_j)} K_j(K_i(s)) = w$, with $0 \leq i < j$.

(a) If $i \leq \ell$, then we have $u \longrightarrow^{(K_i S_{\ell+1})} S_{\ell+2}(K_i(K_\ell(K_{j-1}(s))), K_i(I_\ell)) \longrightarrow^{(K_i K_{\ell+1}),(K_i K_j),(K_i I_{\ell+1})} S_{\ell+2}(K_{\ell+1}(K_j(K_i(s))), I_{\ell+1}) \longrightarrow^{(\eta S_{\ell+1})} K_j(K_i(s)) = w$.

(b) If $i > \ell$, then $u \longrightarrow^{(K_\ell K_j)} K_i(S_{\ell+1}(K_j(K_\ell(s)), I_\ell))$ (because $\ell \leq i < j$) $\longrightarrow^{(S_{\ell+1} K_j)} K_i(K_{j-1}(S_{\ell+1}(K_\ell(s), I_\ell)))$ (because $\ell < i < j$, so $\ell + 1 < j$) $\longrightarrow^{(K_i K_j)} K_j(K_i(S_{\ell+1}(K_\ell(s), I_\ell))) \longrightarrow^{(\eta S_\ell)} K_j(K_i(s)) = w$.

6. $u = K_i(S_{\ell+1}(K_\ell(S_j(s,t)), I_\ell)) \longrightarrow^{(\eta S_\ell)} v = K_i(S_j(s,t)) \longrightarrow^{(K_i S_j)} S_{j+1}(K_i(s), K_i(t)) = w$, with $0 \leq i < j$.

(a) If $i \leq \ell$, then we have $u \longrightarrow^{(K_i S_{\ell+1})} S_{\ell+2}(K_i(K_\ell(S_j(s,t))), K_i(I_\ell)) \longrightarrow^{(K_i K_{\ell+1}),(K_i S_j),(K_i I_{\ell+1})} S_{\ell+2}(K_{\ell+1}(S_{j+1}(K_i(s), K_i(t))), I_{\ell+1}) \longrightarrow^{(\eta S_{\ell+1})} S_{j+1}(K_i(s), K_i(t)) = w$.

(b) If $i > \ell$, then we get the reduction: $u \overset{(K_\ell S_j)}{\longrightarrow} K_i(S_{\ell+1}(S_{j+1}(K_\ell(s), K_\ell(t)), I_\ell))$ (this rule is indeed applicable since $\ell \leq i < j$) $\longrightarrow^{(S_{\ell+1} S_{j+1})} K_i(S_j(S_{\ell+1}(K_\ell(s), I_\ell), S_{\ell+1}(K_\ell(t), I_\ell))) \longrightarrow^{(K_i S_j)} S_{j+1}(K_i(S_{\ell+1}(K_\ell(s), I_\ell)), K_i(S_{\ell+1}(K_\ell(t), I_\ell))) \longrightarrow^{(\eta S_\ell)} S_{j+1}(K_i(s), K_i(S_{\ell+1}(K_\ell(t), I_\ell))) \longrightarrow^{(\eta S_\ell)} S_{j+1}(K_i(s), K_i(t)) = w$.

7. $u = S_i(S_{\ell+1}(K_\ell(K_i(s)), I_\ell), t) \longrightarrow^{(\eta S_\ell)} v = S_i(K_i(s), t) \longrightarrow^{(SK_i)} s = w$.

(a) If $i < \ell$, then we have $u \longrightarrow^{(S_i S_{\ell+1})} S_\ell(S_i(K_\ell(K_i(s)), t), S_i(I_\ell, t)) \longrightarrow^{(S_i K_\ell),(SK_i),(S_i I_\ell)} S_\ell(K_{\ell-1}(s), I_{\ell-1}) \longrightarrow^{(\eta S_{\ell-1})} s = w$.

(b) If $i > \ell$, then we have $u \overset{(K_\ell K_{i+1})}{\longrightarrow} S_i(S_{\ell+1}(K_{i+1}(K_\ell(s)), I_\ell), t) \longrightarrow^{(S_{\ell+1} K_{i+1})} S_i(K_i(S_{\ell+1}(K_\ell(s), I_\ell)), t) \longrightarrow^{(SK_i)} S_{\ell+1}(K_\ell(s), I_\ell) \longrightarrow^{(\eta S_\ell)} s = w$.

(c) If $i = \ell$, then $u = S_\ell(S_{\ell+1}(K_\ell(K_\ell(s)), I_\ell), t) \longrightarrow^{(S_\ell S_{\ell+1})} S_\ell(S_\ell(K_\ell(K_\ell(s)), t), S_\ell(I_\ell, t)) \longrightarrow^{(SK_\ell)} S_\ell(K_\ell(s), S_\ell(I_\ell, t)) \longrightarrow^{(SK_\ell)} s = w$.

8. $u = S_i(S_{\ell+1}(K_\ell(I_i), I_\ell), t) \longrightarrow^{(\eta S_\ell)} v = S_i(I_i, t) \longrightarrow^{(SI_i)} t = w$.

(a) If $i < \ell$, then $u \longrightarrow^{(S_i S_{\ell+1})} S_\ell(S_i(K_\ell(I_i), t), S_i(I_\ell, t)) \longrightarrow^{(S_i K_\ell),(SI_i),(S_i I_\ell)} S_\ell(K_{\ell-1}(t), I_{\ell-1}) \longrightarrow^{(\eta S_{\ell-1})} t = w$.

(b) If $i > \ell$, then $u \longrightarrow^{(K_\ell I_{i+1})} S_i(S_{\ell+1}(I_{i+1} I_\ell, t), \longrightarrow)^{(S_{\ell+1} I_{i+1})} S_i(I_i, t) \longrightarrow^{(SI_i)} t = w$.

(c) If $i = \ell$, then $u = S_\ell(S_{\ell+1}(K_\ell(I_\ell), I_\ell), t) \longrightarrow^{(S_\ell S_{\ell+1})} S_\ell(S_\ell(K_\ell(I_\ell), t), S_\ell(I_\ell, t)) \longrightarrow^{(SK_\ell)} S_\ell(I_\ell, S_\ell(I_\ell, t)) \longrightarrow^{(SI_\ell)} S_\ell(I_\ell, t) \longrightarrow^{(SI_\ell)} t = w$.

It follows that the SKInT-strongly normalizing terms are exactly the SKInT$_\eta$-strongly normalizing terms.

We now show that for every term u, u is SKInT-solvable if and only if it is SKInT$_\eta$-solvable. Call a term *spine-normal* if it has no redex on its spine.

We first claim that: (b) if u_1 is spine-normal in SKInT, and u_1 rewrites to u_2 by η on the spine, then u_2 is also spine-normal in SKInT. Notice indeed that for u_2 not to be SKInT-spine-normal, it would need to contain a spine-redex $S_\ell(v, w)$ or $K_\ell(v)$ (so that in each case the degree $d(v)$ is at least $\ell + 1$; see item (*vii*) in Appendix D for the notion of degree) coming from an η-redex $S_\ell(S_{j+1}(K_j(v), I_j), w)$, resp. $K_\ell(S_{j+1}(K_j(v), I_j))$ in u_1. Since u_1 is SKInT-spine-normal, $\ell \geq j + 1$ in each case, so $d(v) \geq j + 2$: but then u_1 cannot be SKInT-spine-normal, in both cases.

It follows that: (c) if u is SKInT-solvable, then u is SKInT$_\eta$-solvable. Indeed, let v be a SKInT-spine-normal form of u. Then reduce all the η-redexes on the spine of v (this terminates, because η decreases the size of terms): by (b), the resulting normal form is SKInT$_\eta$-spine-normal.

Also: (d) if u is weakly normalizing in SKInT, then it is also weakly normalizing in SKInT$_\eta$. Indeed, we claim that: (e) if u_1 is SKInT-normal and u_1 rewrites to u_2 by the η rule, then u_2 is SKInT-normal as well. Then (d) follows from (e): just normalize u in SKInT to get a normal form v, then apply η until a SKInT$_\eta$-normal form is reached (η indeed clearly terminates). To prove (e), notice that the only possibility for u_2 not to be SKInT-normal is for u_2 to contain a redex $S_\ell(v, w)$ or $K_\ell(v)$ (so that in each case the degree $d(v)$ is at least $\ell + 1$; see item (*vii*) in Appendix D for the notion of degree) coming from an η-redex $S_\ell(S_{j+1}(K_j(v), I_j), w)$, resp. $K_\ell(S_{j+1}(K_j(v), I_j))$ in u_1. Since u_1 is normal, $\ell \geq j + 1$ in each case, so $d(v) \geq j + 2$: but then u_1 cannot be SKInT-normal, in both cases.

References

[ACCL90] Martín Abadi, Luca Cardelli, Pierre-Louis Curien, and Jean-Jacques Lévy. Explicit substitutions. In *Proceedings of the 17th Annual ACM Symposium on Principles of Programming Languages*, pages 31–46, San Francisco, California, January 1990.

[Bar84] Henk 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.

[BdP92] Gavin Bierman and Valeria de Paiva. Intuitionistic necessity revisited. In *Logic at Work*, Amsterdam, the Netherlands, 1992.

[CC90] Felice Cardone and Mario Coppo. Two extensions of Curry's type inference system. In P. Odifreddi, editor, *Logic and Computer Science*, volume 31 of *The APIC Series*, pages 19–76. Academic Press, 1990.

[CF58] Haskell B. Curry and Robert Feys. *Combinatory Logic*, volume 1. North-Holland, 1958.

[Cur86] Pierre-Louis Curien. *Categorical Combinators, Sequential Algorithms and Functional Programming*. Pitman, London, 1986.

[Der87] Nachum Dershowitz. Termination of rewriting. *Journal of Symbolic Computation*, 3:69–116, 1987.

[GGL98] Healfdene Goguen and Jean Goubault-Larrecq. Sequent combinators: A Hilbert system for the lambda calculus. 1998. To be submitted; available at `http://www.dyade.fr/fr/actions/vip/jgl/skinshort.ps.gz`.

[GL96] Jean Goubault-Larrecq. On computational interpretations of the modal logic S4 I. Cut elimination. Technical report, University of Karlsruhe, 1996. Available on `ftp://theory.doc.ic.ac.uk/theory/guests/GoubaultJ/`.

[GL97] Jean Goubault-Larrecq. On computational interpretations of the modal logic S4 IIIb. Confluence and conservativity of the $\lambda\mathtt{ev}Q_H$-calculus. Technical report, Inria, 1997.

[GLM97] Jean Goubault-Larrecq and Ian Mackie. *Proof Theory and Automated Deduction*, volume 6 of *Applied Logic Series*. Kluwer, May 1997. ISBN 0-7923-4593-2.

[HL89] Thérèse Hardin and Jean-Jacques Lévy. A confluent calculus of substitutions. In *France-Japan Artificial Intelligence and Computer Science Symposium*, December 1989.

[LRD94] Pierre Lescanne and Jocelyne Rouyer-Degli. From $\lambda\sigma$ to $\lambda\upsilon$: a journey through calculi of explicit substitutions. In *Proceedings of the 21st Annual ACM Symposium on Principles of Programming Languages*, 1994.

[Mel95] Paul-André Melliès. Typed lambda-calculi with explicit substitutions may not terminate. In M. Dezani-Ciancaglini and G. Plotkin, editors, *2nd International Conference on Typed Lambda-Calculi and Applications (TLCA'95)*, pages 328–334, Edinburgh, UK, April 1995. Springer Verlag LNCS 902.

[MH96] César Augusto Muñoz Hurtado. Confluence and preservation of strong normalization in an explicit substitutions calculus. In *Proceedings of the 11th ACM/IEEE Symposium on Logics in Computer Science*, 1996. Long version available as INRIA Research Report 2762, December 1995.

[Plo75] Gordon Plotkin. Call-by-name, call-by-value and the λ-calculus. *Theoretical Computer Science*, 1(2):125–159, 1975.

[Say97] Émilie Sayag. *Types intersections simples*. PhD thesis, Université Paris VII, 1997.

Unité de recherche INRIA Lorraine, Technopôle de Nancy-Brabois, Campus scientifique,
615 rue du Jardin Botanique, BP 101, 54600 VILLERS LÈS NANCY
Unité de recherche INRIA Rennes, Irisa, Campus universitaire de Beaulieu, 35042 RENNES Cedex
Unité de recherche INRIA Rhône-Alpes, 46 avenue Félix Viallet, 38031 GRENOBLE Cedex 1
Unité de recherche INRIA Rocquencourt, Domaine de Voluceau, Rocquencourt, BP 105,
78153 LE CHESNAY Cedex
Unité de recherche INRIA Sophia-Antipolis, 2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex

Éditeur
INRIA, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
ISSN 0249-6399