UNIVERSIDAD DE BUENOS AIRES
Facultad de Ciencias Exactas y Naturales
Departamento de Computación

Tesis de Licenciatura

Factorización de derivaciones a través de tipos intersección
Factorizing derivations via intersection types

Gonzalo Ciruelos Rodríguez

Director: Pablo Barenbaum

Fecha de Presentación: ??

En sistemas de tipos intersección no idempotentes típicos, la normalización de pruebas no es confluente. En este trabajo presentamos un sistema confluente de tipos intersección no idempotentes para el cálculo λ. Escribimos las derivaciones de tipos usando una sintáxis concisa de términos de prueba. El sistema goza de buenas propiedades: *subject reduction*, es fuertemente normalizante, y tiene una teoría de residuos muy regular. Establecemos una correspondencia con el cálculo lambda mediante teoremas de simulación.

La maquinaria de los tipos intersección no idempotentes nos permite seguir el rastro del uso de los recursos necesarios para obtener una respuesta. En particular, induce una noción de *basura*: un cómputo es basura si no contribuye a hallar una respuesta. Usando estas nociones, mostramos que el espacio de derivaciones de un término λ puede ser factorizado usando una variante de la construcción de Grothendieck para semireticulados. Esto significa, en particular, que cualquier derivación del cálculo λ puede ser escrita de una única manera como un prefijo libre de basura, seguido de basura.

Palabras clave: Cálculo lambda, Tipos intersección, Espacio de derivacion, Reticulado

In typical non-idempotent intersection type systems, proof normalization is not confluent. In this work we introduce a confluent non-idempotent intersection type system for the λ-calculus. Typing derivations are presented using a concise proof term syntax. The system enjoys good properties: subject reduction, strong normalization, and a very regular theory of residuals. A correspondence with the λ-calculus is established by simulation theorems.

The machinery of non-idempotent intersection types allows us to track the usage of resources required to obtain an answer. In particular, it induces a notion of *garbage*: a computation is garbage if it does not contribute to obtaining an answer. Using these notions, we show that the derivation space of a λ-term may be factorized using a variant of the Grothendieck construction for semilattices. This means, in particular, that any derivation in the λ-calculus can be uniquely written as a garbage-free prefix followed by garbage.

Keywords: Lambda Calculus, Intersection Types, Derivation Space, Lattice

Contents

Introduction

The main concern of this work will be to analyze and understand *spaces of computations*. To understand what a space of computations is, we need to introduce the notion of rewriting systems. A rewriting system is a set of objects A and a binary *rewriting* relation on A, called $\to$. If $a_1 \to a_2$ we say that a_1 can be *rewritten* to a_2.

For example, the elementary arithmetic (addition and multiplication of natural numbers) can be seen as a rewriting system, where the objects (called *terms* in this case) are:

$$t, s ::= t + s \mid t \cdot s \mid \underline{n}$$

with $n \in \mathbb{N}$. We write $\underline{n}$ to mean that we have a term, not only a natural number. The terms are equipped with three rewriting rules, one that says how to add and two that say how to multiply:

$$\begin{array}{rcll} \underline{n} + \underline{m} & \to & \underline{n+m} & \\ \underline{n} \cdot t & \to & \underbrace{t + t + ... + t}_{n \text{ times}} & \text{if } n > 0 \\ \underline{0} \cdot t & \to & \underline{0} & \end{array}$$

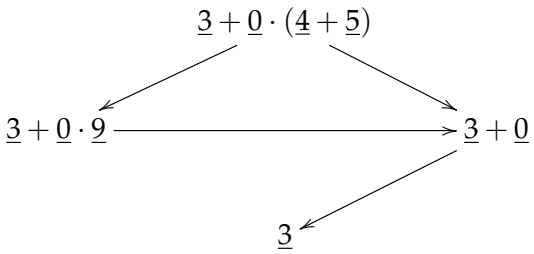
Actually, to be completely formal we should say that the rewriting relation is the *contextual closure* of the rules above, which intuitively means that we can apply the rules anywhere we want in the term. For example, if we had the term $\underline{3} + \underline{0} \cdot (\underline{4} + \underline{5})$, we can apply the first rule to get $\underline{3} + \underline{0} \cdot \underline{9}$, and then the third one to get $\underline{3} + \underline{0}$, which can be rewritten to $\underline{3}$. A derivation is simply a sequence of composable steps.

But notice that instead of doing the first step we did, we could have directly applied the third rewrite rule onto our term, which would have yielded $\underline{3} + \underline{0}$, saving one step.

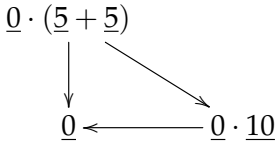
Given a term t, its *reduction graph* is the graph that has all reducts of t as nodes, and edges between two nodes if one can be rewritten to the other.

We could also consider the set of all reductions that start in t, and equip it with an order relation—usually a derivation is *less than* another derivation if it does less work. The *space of computations* of t is that set modulo an equivalence relation—which usually says whether or not two derivations do the same amount of work. We will use the same kind of diagrams to represent computation spaces and reduction graphs.

In the example above, the computation space of $\underline{3} + \underline{0} \cdot (\underline{4} + \underline{5})$ would be the following.

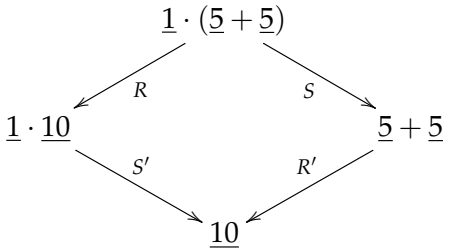


To learn some other notions of rewriting theory, consider the following examples.



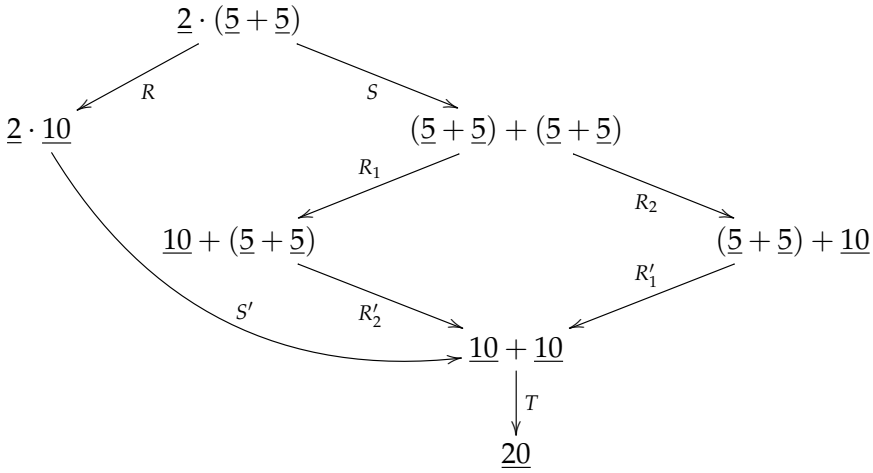
Notice how the step that performed the sum was not needed, as we could have done the multiplication directly. We will concentrate in these kinds of phenomena in this work: steps that do not perform interesting work. Later on, we will call these steps *garbage*.

In the next example we will name our steps to make the explanation clearer.



This example allows us to present the notion of *residuals*. We have two steps that perform the sum $\underline{5} + \underline{5} \to \underline{10}$, R and R'. These steps are formally different (they are different elements of the rewriting relation), but do the same work. We can formalize this notion: we will say that R' is the residual of R after S, *i.e.*, it is what is left of R after S. We will write this $R/S = \{R'\}$. Also notice how there is no "faster" reduction strategy, all reductions are the same length.

Let us consider a last, slightly more complicated, example.



Notice here how the residual of R after S is a set of two steps, $\{R_1, R_2\}$. That means that if we performed S, we need to do those two steps to do the work R alone did.

We would be inclined to guess that a good reduction strategy would be to always multiply a number to a number, such that we don't multiply work. But that strategy fails when we have a 0 as an operand.

What we want to do in this work is to understand the spaces of computations of programs, which may have a very complex structure. Consider a side-effect free programming language. The possible computations of a tuple (A, B) are rewrite sequences $(A, B) \to \ldots \to (A', B')$. These sequences can always be decomposed as two non-interfering computations $A \to \ldots \to A'$ and $B \to \ldots \to B'$. The reason is that the sub-expressions A and B cannot interact with each other. Indeed, the space of computations of (A, B) can be understood as the *product* of the spaces of A and B. In contrast, the space of computations of a function application $f(A)$ is not so easy to characterize. The difficulty is that f may use the value of A zero, one, or possibly many times.

Another difficulty is that a program may *hang*, which means it has an infinite rewrite sequence. This did not happen in the elementary arithmetic example above, where we always arrive to a result. More importantly, a program may finish or not depending on what rewriting rule we choose in each step.

We hope understanding the spaces of computations of programs may be helpful to better understand the properties of *evaluation strategies*, such as call-by-name or call-by-value, from a quantitative point of view. A better understanding may also suggest program optimizations, and it should allow to justify that certain program conversions are sound: *e.g.* that they do not turn a terminating program into a non-terminating one.

The study of different notions of equivalence of programs [JHM69, BMPR16] may greatly benefit from a better understanding of space of computations, too. Usually, several different notions of observational equivalence may be established, depending of how a program behaves under an arbitrary context `C`. For example, a possible definition says that two terms M and N are observationally equivalent if for any context `C`, we have

that $\mathtt{C}\langle M\rangle$ has a normal form if and only if $\mathtt{C}\langle N\rangle$ has a normal form. Understanding the space of computations of M (and N) would be a key tool to reason about the different notions of observational equivalence.

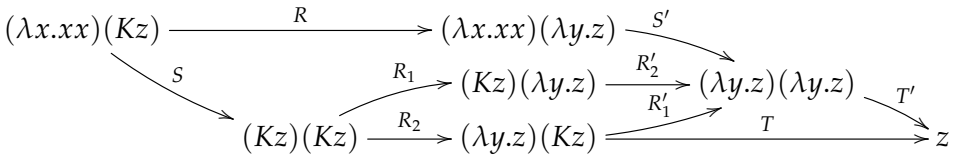
The pure λ-calculus is the quintessential functional programming language. The λ-calculus is a rewriting system that can represent all computable functions in a concise manner. Terms of the λ-calculus can be variables, functions or applications.

$$t, s ::= x \mid \lambda x.t \mid t\, s$$

And we only have one rewriting rule, the function application $(\lambda x.t)s \to t\{x := s\}$, where $t\{x := s\}$ means to replace every occurrence of x in t with s.

Computations in the λ-calculus have been thoroughly studied since its conception in the 1930s. The well-known theorem by Church and Rosser [CR36] states that the λ-calculus is *confluent*, which means, in particular, that terminating programs have unique normal forms. Another result by Curry and Feys [CF58] states that computations in the λ-calculus may be *standardized*, meaning that they may be converted into a computation in canonical form. A refinement of this theorem by Lévy [Lév78] asserts that the canonical computation thus obtained is equivalent to the original one in a strong sense, namely that they are *permutation equivalent*. In a series of papers [Mel97, Mel00, Mel02a, Mel02b, Mel05], Melliès generalized many of these results to the abstract setting of *axiomatic rewrite systems*.

Let us discuss "spaces of computations" more precisely. The *derivation space* of an object x in some rewriting system is the set of all *derivations*, *i.e.* sequences of rewrite steps, starting from x. In this work, our rewriting system of interest will be the pure λ-calculus, and we will be interested in *finite* derivations only. In the λ-calculus, a transitive relation between derivations may be defined, the *prefix order*. A derivation ρ is a prefix of a derivation σ, written $\rho \sqsubseteq \sigma$, whenever ρ performs less computational work than σ. Formally, $\rho \sqsubseteq \sigma$ is defined to hold whenever the *projection* ρ/σ is empty[1]. For example, if $K = \lambda x.\lambda y.x$, the derivation space of the term $(\lambda x.xx)(Kz)$ can be depicted with the *reduction graph* below. Derivations are directed paths in the reduction graph, and ρ is a prefix of σ if there is a directed path from the target of ρ to the target of σ. For instance, SR_2 is a prefix of $RS'T'$:



(1)

Remark that the relation $\sqsubseteq$ is reflexive and transitive but not anti-symmetric, *i.e.* it is a quasi-order but not an order. For example $RS' \sqsubseteq SR_1R'_2 \sqsubseteq RS'$ but $RS' \neq SR_1R'_2$. Anti-symmetry may be recovered as usual when in presence of a quasi-order, namely

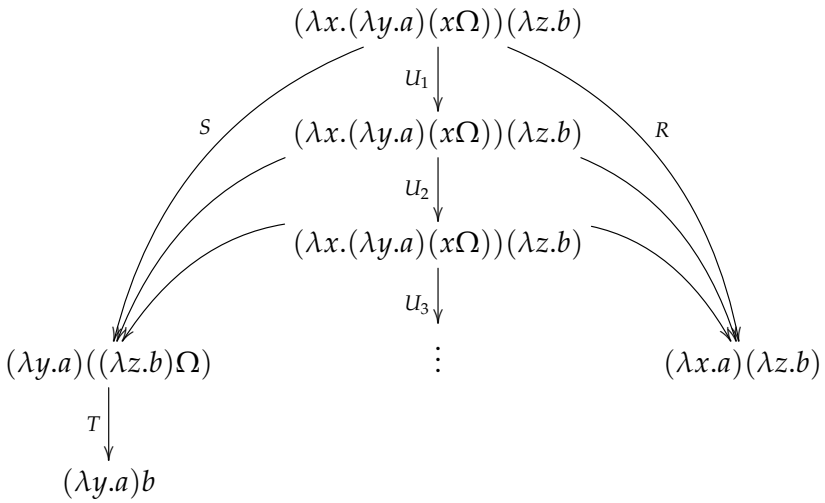
[1] The notion of projection defined by means of residuals is the standard one, see *e.g.* [Bar84, Chapter 12] or [Ter03, Section 8.7]. We will define this more formally in the preliminaries.

Un ref a esa definición (o a la sección) estaría bueno acá

by working modulo *permutation equivalence*: two derivations ρ and σ are said to be permutation equivalent, written $\rho \equiv \sigma$, if $\rho \sqsubseteq \sigma$ and $\sigma \sqsubseteq \rho$. Working modulo permutation equivalence is reasonable because Lévy's formulation of the standardization theorem ensures that permutation equivalence is decidable, and each equivalence class has a canonical representative.

Derivation spaces are known to exhibit various regularities [Lév78, Zil84, Lan94, Mel96, Lev15, AL13]. In his PhD thesis, Lévy [Lév78] showed that the derivation space of a term is an upper semilattice: any two derivations ρ, σ from a term t have a *least upper bound* $\rho \sqcup \sigma$, defined as $\rho(\sigma/\rho)$, unique up to permutation equivalence. On the other hand, the derivation space of a term t is not an easy structure to understand in general[2]. For example, relating the derivation space of an application ts with the derivation spaces of t and s appears to be a hard problem. Lévy also noted that the *greatest lower bound* of two derivations does not necessarily exist, meaning that the derivation space of a term does not form a lattice in general. And even when it forms a lattice, it may not necessarily be a *distributive* lattice, as observed by Laneve [Lan94].

Consider the following counterexample [3], showing that the meet of derivations is not well-defined in general. Let $\Omega = (\lambda x.xx)\lambda x.xx$, and consider the reduction space of $(\lambda x.(\lambda y.a)(x\Omega))(\lambda z.b)$, where the steps U_i contract Ω:



Observe that ST and R do not have a greatest lower bound: ST and R are not comparable, and neither are S and R. Furthermore, for all $n \in \mathbb{N}$ we have that $U_1 \dots U_n \sqsubseteq R$ and $U_1 \dots U_n \sqsubseteq ST$ so the meet $R \sqcap ST$ does not exist.

In [Mel97], Melliès showed that derivation spaces in any rewriting system satisfying certain axioms may be factorized using two spaces, one of *external* and one of *internal* derivations.

The difficulty to understand derivation spaces is due to three pervasive phenomena of *interaction* between computations. The first phenomenon is *duplication*: in the

[2] Problem 2 in the RTA List of Open Problems [DJK91] poses the open-ended question of investigating the properties of "spectra", *i.e.* derivation spaces.

[3] Similar to an example due to Laneve [Lan94].

reduction graph above (1), the step S duplicates the step R, resulting in two copies of R: the steps R_1 and R_2. In such situation, one says that R_1 and R_2 are *residuals* of R, and, conversely, R is an *ancestor* of R_1 and R_2. The second phenomenon is *erasure*: in the diagram above, the step T erases the step R'_1, resulting in no copies of R'_1. The third phenomenon is *creation*: in the diagram above, the step R_2 creates the step T, meaning that T is not a residual of a step that existed prior to executing R_2; that is, T has no ancestor.

in the same graph

These three interaction phenomena, especially duplication and erasure, are intimately related with the management of *resources*. In this work, we aim to explore the hypothesis that **having an explicit representation of resource management may provide insight on the structure of derivation spaces**.

There are many existing λ-calculi that deal with resource management explicitly [Bou93, ER03, KL07, KR], most of which draw inspiration from Girard's Linear Logic [Gir87]. In recent years, one family of such formalisms, namely calculi endowed with *non-idempotent intersection type systems*, has received some attention [Ehr12, BL13, BKDR14, BKV17, Kes16, Via17, KRV18]. These type systems are able to statically capture non-trivial dynamic properties of terms, particularly *normalization*, while at the same time being amenable to elementary proof techniques by induction, rather than arguments based on reducibility. Intersection types were originally proposed by Coppo and Dezani-Ciancaglini [CD78] to study termination in the λ-calculus. They are characterized by the presence of an *intersection* type constructor $\tau \cap \sigma$. *Non-idempotent* intersection type systems are distinguished from their usual idempotent counterparts by the fact that intersection is not declared to be idempotent, *i.e.* τ and $\tau \cap \tau$ are not equivalent types. Rather, intersection behaves like a multiplicative connective in linear logic. Arguments to functions are typed many times, typically once per each time that the argument will be used. Non-idempotent intersection types were originally formulated by Gardner [Gar94], and later reintroduced by de Carvalho [Car07].

In this work, we will use a non-idempotent intersection type system based on system $\mathcal{W}$ of [BKV17] (called system $\mathcal{H}$ in [BKDR14]). Let us recall its definition. Terms are as usual in the λ-calculus ($t ::= x \mid \lambda x.t \mid t\,t$). Types $\tau, \sigma, \rho, \ldots$ are defined by the grammar:

$$\tau \quad ::= \quad \alpha \mid \mathcal{M} \to \tau \qquad\qquad \mathcal{M} \quad ::= \quad [\tau_i]_{i=1}^n \quad \text{with } n \geq 0$$

where α ranges over one of denumerably many *base types*, and $\mathcal{M}$ represents a *multiset of types*. Here $[\tau_i]_{i=1}^n$ denotes the multiset containing the types $\tau_1, \ldots, \tau_n$ with their respective multiplicities. A multiset $[\tau_i]_{i=1}^n$ intuitively stands for the (non-idempotent) intersection $\tau_1 \cap \ldots \cap \tau_n$, and $\oplus$ stands for disjoint union. Type assignment rules for system $\mathcal{W}$ are as follows.

Definition 0.1 (System $\mathcal{W}$).

$$\frac{}{x : [\tau] \vdash \tau}\ \text{var} \qquad \frac{\Gamma \oplus (x : \mathcal{M}) \vdash t : \tau}{\Gamma \vdash \lambda x.t : \mathcal{M} \to \tau}\ \to_{\mathrm{I}} \qquad \frac{\Gamma \vdash t : [\sigma_i]_{i=1}^n \to \tau \quad (\Delta_i \vdash s : \sigma_i)_{i=1}^n}{\Gamma \oplus \bigoplus_{i=1}^n \Delta_i \vdash t\,s : \tau}\ \to_{\mathrm{E}}$$

x

¿Esta notación se define recién en la Sección 1.4

A esta altura, está dificil de entender la notación

Estaría bueno enumerarlas

Observe that the $\to_{\mathrm{E}}$ rule has $n+1$ premises, where $n \geq 0$. System $\mathcal{W}$ enjoys various properties, nicely summarized in [BKV17].

There are two obstacles to adopting system $\mathcal{W}$ for studying derivation spaces. The first obstacle is just a matter of presentation—typing derivations use a tree-like notation, which is cumbersome. One would like to have an alternative presentation based on proof terms. For example, one would like to write x^{τ} for an application of the `var` rule, $\lambda x.t$ for an application of the $\to_{\mathrm{I}}$ rule, and $t[s_1, \ldots, s_n]$ for an application of the $\to_{\mathrm{E}}$ rule, so that, for example, $\lambda x.x^{[\alpha,\alpha]\to\beta}[x^{\alpha}, x^{\alpha}]$ represents the following typing derivation:

$$\dfrac{\dfrac{\dfrac{}{x : [\alpha,\alpha] \to \beta \vdash x : [\alpha,\alpha] \to \beta}\,\mathtt{var} \quad \dfrac{}{x : [\alpha] \vdash x : \alpha}\,\mathtt{var} \quad \dfrac{}{x : [\alpha] \vdash x : \alpha}\,\mathtt{var}}{x : [[\alpha,\alpha] \to \beta, \alpha, \alpha] \vdash xx : \beta}\,\to_{\mathrm{E}}}{\vdash \lambda x.xx : [[\alpha,\alpha] \to \beta, \alpha, \alpha] \to \beta}\,\to_{\mathrm{I}}$$

The second obstacle is a major one for our purposes: *proof normalization* in this system is not confluent. The reason is that applications take multiple arguments, and a β-reduction step must choose a way to distribute these arguments among the occurrences of the formal parameters. For instance, the following critical pair cannot be closed:

No se entiende

$$(\lambda x.y^{[\alpha]\to[\alpha]\to\beta}[x^{\alpha}][x^{\alpha}])[z^{[\gamma]\to\alpha}[z^{\gamma}], z^{[]\to\alpha}[]] \to y^{[\alpha]\to[\alpha]\to\beta}[z^{[\gamma]\to\alpha}[z^{\gamma}]][z^{[]\to\alpha}[]]$$
$$(\lambda x.y^{[\alpha]\to[\alpha]\to\beta}[x^{\alpha}][x^{\alpha}])[z^{[\gamma]\to\alpha}[z^{\gamma}], z^{[]\to\alpha}[]] \to y^{[\alpha]\to[\alpha]\to\beta}[z^{[]\to\alpha}[]][z^{[\gamma]\to\alpha}[z^{\gamma}]]$$

The remainder of this work is organized as follows:

- In Chapter 1, we review some standard notions of order and rewriting theory, as well as some basic notions of the λ-calculus that we will use throughout the work.
- In Chapter 2, we introduce a confluent calculus, $\lambda^{\#}$, based on system $\mathcal{W}$. The desirable properties of system $\mathcal{W}$ of [BKV17] still hold in $\lambda^{\#}$. Moreover, $\lambda^{\#}$ is confluent. We impose confluence forcibly, by decorating sub-trees with distinct labels, so that a β-reduction step may distribute the arguments in a unique way.
- In Chapter 3, we develop a theory of residuals for the $\lambda^{\#}$-calculus and prove that the derivation spaces of its terms have an orderly structure, namely they are distributive lattices.
- In Chapter 4, we establish a correspondence between derivation spaces in the λ-calculus and the $\lambda^{\#}$-calculus via simulation theorems, which defines a morphism of upper semilattices.
- In Chapter 5, we introduce the notion of a garbage derivation. Roughly, a derivation in the λ-calculus is *garbage* if it maps to an empty derivation in the $\lambda^{\#}$-calculus. This gives rise to an orthogonal notion of *garbage-free* derivation. The notion of garbage-free derivation is closely related with the notions of *needed step* [Ter03, Section 8.6], *typed occurrence of a redex* [BKV17], and *external* derivation [Mel97].

Using this notion of garbage we prove a *factorization theorem* reminiscent of Melliès' [Mel97]. The upper semilattice of derivations of a term in the λ-calculus is factorized using a variant of the Grothendieck construction. Every derivation is uniquely decomposed as a garbage-free prefix followed by a garbage suffix.

- In Conclusions, we end with a discussion of our results.

Note. Proofs including a ♠ symbol are spelled out in detail in the appendix.

Chapter 1

Preliminaries

1.1 Order theory

We are interested in understanding the derivation spaces of λ-terms. These derivation spaces, as we briefly mentioned earlier, have an structure that can be seen as an order. More specifically, the poset we will consider will be the one of derivations, where the order is giving by the amount of *work* each derivation does.

But the structures we will work with are more than just posets, they have a richer structure, some of which we will define now.

An **upper semilattice** is a poset (*i.e.*, a set A with an order $\leq$) with a least element or **bottom** $\bot \in A$, such that for every two elements $a, b \in A$ there is a least upper bound or **join** $(a \vee b) \in A$.

A **lattice** is an upper semilattice with a greatest element or **top** $\top \in A$, and such that for every two elements $a, b \in A$ there is a greatest lower bound or **meet** $(a \wedge b) \in A$. A lattice is **distributive** if $\wedge$ distributes over $\vee$ and vice versa.

In the introduction we claimed that derivation spaces of λ-terms where upper semilattices, but in general were not lattices.

A **morphism** of upper semilattices is given by a monotonic function $f : A \to B$, *i.e.* $a \leq b$ implies $f(a) \leq f(b)$, preserving the bottom element, *i.e.* $f(\bot) = \bot$, and joins, *i.e.* $f(a \vee b) = f(a) \vee f(b)$ for all $a, b \in A$. Similarly for morphisms of lattices (and distributive lattices).

Any poset $(A, \leq)$ may be regarded as category whose objects are the elements of A and morphisms are of the form $a \hookrightarrow b$ for all $a \leq b$. The category of posets with monotonic functions is denoted by Poset. In fact, we view it as a 2-category: given morphisms $f, g : A \to B$ of posets, there is a *2-cell* $f \leq g$ if $f(a) \leq g(a)$ for all $a \in A$. (A 2-category is just a category with morphisms between morphisms). This notion is more technical and will only be used in the last chapter.

1.2 Rewriting theory

The λ-calculus is a particular case of a more general mathematical concept, called **rewriting system**. Informally, a rewriting system is a set of objects and a set of rules that let you transform some objects into others.

More formally, an **axiomatic rewrite system** (*cf.* [Mel96, Def. 2.1]) is given by a set of objects Obj, a set of steps Stp, two functions $\mathsf{src}, \mathsf{tgt} : \mathsf{Stp} \to \mathsf{Obj}$ indicating the source and target of each step, and a **residual function** $(/)$ such that given any two steps $R, S \in \mathsf{Stp}$ with the same source, yields a set of steps R/S such that $\mathsf{src}(R') = \mathsf{tgt}(S)$ for all $R' \in R/S$. Steps are ranged over by $R, S, T, \ldots$. A step $R' \in R/S$ is called a **residual** of R after S, and R is called an **ancestor** of R'. Steps are **coinitial** (resp. **cofinal**) if they have the same source (resp. target). A **derivation** is a possibly empty sequence of composable steps $R_1 \ldots R_n$. Derivations are ranged over by $\rho, \sigma, \tau, \ldots$. The functions src and tgt are extended to derivations (noting that there is a different empty derivation for each element in Obj). We use ϵ_t to denote the empty derivation with source (and target) t, and we will often drop the subscript when it is clear from the context.

noticing

Composition of derivations is defined when $\mathsf{tgt}(\rho) = \mathsf{src}(\sigma)$ and written $\rho\sigma$. Residuals are extended for projecting after a derivation, namely $R_n \in R_0/S_1 \ldots S_n$ if and only if there exist $R_1, \ldots, R_{n-1}$ such that $R_{i+1} \in R_i/S_{i+1}$ for all $0 \leq i \leq n-1$. Let $\mathcal{M}$ be a set of coinitial steps.

A **development** of $\mathcal{M}$ is a (possibly infinite) derivation $R_1 \ldots R_n \ldots$ such that for every index i there exists a step $S \in \mathcal{M}$ such that $R_i \in S/R_1 \ldots R_{i-1}$. Informally, a development of $\mathcal{M}$ is a derivation in which we only do the work that the steps contained in $\mathcal{M}$ do. A development is **complete** if it is maximal.

The definition of an abstract rewriting system is very general, and because of that it does not give us general properties for systems with such structure. In his PhD thesis, Melliès gave a set of sufficient properties (which he called axioms) that a rewriting system should have to behave properly.

An **orthogonal** axiomatic rewrite system (*cf.* [Mel96, Sec. 2.3]) has four additional axioms[1]:

1. *Autoerasure*. $R/R = \varnothing$ for all $R \in \mathsf{Stp}$.
2. *Finite Residuals*. The set R/S is finite for all coinitial $R, S \in \mathsf{Stp}$.
3. *Finite Developments*. If $\mathcal{M}$ is a set of coinitial steps, all developments of $\mathcal{M}$ are finite.
4. *Semantic Orthogonality*. Let $R, S \in \mathsf{Stp}$ be coinitial steps. Then there exist a complete development ρ of R/S and a complete development σ of S/R such that ρ and σ are cofinal. Moreover, for every step $T \in \mathsf{Stp}$ such that T is coinitial to R, the following equality between sets holds: $T/R\sigma = T/S\rho$.

In [Mel96], Melliès develops the theory of orthogonal axiomatic rewrite systems. A notion of **projection** ρ/σ may be defined between coinitial derivations, essentially by setting $\epsilon/\sigma \stackrel{\text{def}}{=} \epsilon$ and $R\rho'/\sigma \stackrel{\text{def}}{=} (R/\sigma)(\rho'/(\sigma/R))$ where, by abuse of notation, R/σ

[1] In [Mel96], Autoerasure is called *Petit axiome A*, Finite Residuals is called *Petit axiome B*, and Semantic Orthogonality is called *Axiome PERM*. We follow the nomenclature of [ABKL14]

stands for a (canonical) complete development of the set R/σ. [2] Using this notion, one may define a transitive relation of **prefix** ($\rho \sqsubseteq \sigma$), a **permutation equivalence** relation ($\rho \equiv \sigma$), and the **join** of derivations ($\rho \sqcup \sigma$). Some of their properties are summed up in the figure below:

Summary of properties of orthogonal axiomatic rewrite systems

$\epsilon\rho = \rho$	$\rho \sqsubseteq \sigma \overset{\text{def}}{\iff} \rho/\sigma = \epsilon$	$\rho \sqsubseteq \sigma \implies \rho/\tau \sqsubseteq \sigma/\tau$
$\rho\epsilon = \rho$	$\rho \equiv \sigma \overset{\text{def}}{\iff} \rho \sqsubseteq \sigma \wedge \sigma \sqsubseteq \rho$	$\rho \sqsubseteq \sigma \iff \tau\rho \sqsubseteq \tau\sigma$
$\epsilon/\rho = \epsilon$	$\rho \sqcup \sigma \overset{\text{def}}{=} \rho(\sigma/\rho)$	$\rho \sqcup \sigma \equiv \sigma \sqcup \rho$
$\rho/\epsilon = \rho$	$\rho \equiv \sigma \implies \tau/\rho = \tau/\sigma$	$(\rho \sqcup \sigma) \sqcup \tau = \rho \sqcup (\sigma \sqcup \tau)$
$\rho/\sigma\tau = (\rho/\sigma)/\tau$	$\rho \sqsubseteq \sigma \iff \exists\tau.\, \rho\tau \equiv \sigma$	$\rho \sqsubseteq \rho \sqcup \sigma$
$\rho\sigma/\tau = (\rho/\tau)(\sigma/(\tau/\rho))$	$\rho \sqsubseteq \sigma \iff \rho \sqcup \sigma \equiv \sigma$	$(\rho \sqcup \sigma)/\tau = (\rho/\tau) \sqcup (\sigma/\tau)$
$\rho/\rho = \epsilon$		

Let $[\rho] = \{\sigma \mid \rho \equiv \sigma\}$ denote the permutation equivalence class of ρ. In an orthogonal axiomatic rewrite system, the set $\mathbb{D}(x) = \{[\rho] \mid \mathsf{src}(\rho) = x\}$ forms an upper semilattice. The order $[\rho] \sqsubseteq [\sigma]$ is given by $\rho \sqsubseteq \sigma$, the join is $[\rho] \sqcup [\sigma] = [\rho \sqcup \sigma]$, and the bottom is $\bot = [\epsilon]$.

The λ-calculus is an example of an orthogonal axiomatic rewrite system [Mel96]. Our structures of interest are the semilattices of derivations of the λ-calculus, written $\mathbb{D}^\lambda(t)$ for any given λ-term t. As usual, β-reduction in the λ-calculus is written $t \to_\beta s$ and defined by the contextual closure of the axiom $(\lambda x.t)s \to_\beta t\{x := s\}$.

1.3 Lists and sets

Throughout this work we will use lists and sets, so we establish here basic definitions and notations.

Definition 1.1 (Lists and sets)**.** If A is a sort, we write $\vec{A}$ for the sort of (finite) **lists** over A, defined inductively as:

$$\vec{A} ::= \epsilon \mid A \cdot \vec{A}$$

We usually write $[a_1, \ldots, a_n]$, abbreviated $[a_i]_{i=1}^n$, to stand for $a_1 \cdot (a_2 \cdot \ldots (a_n \cdot \epsilon))$. If $\vec{a}$ and $\vec{b}$ are lists, $\vec{a} + \vec{b}$ stands for its concatenation, and $|\vec{a}|$ is the length of the list $\vec{a}$. If $a, b, c, \ldots$ are the names of the meta-variables ranging over a sort A, then $\vec{a}, \vec{b}, \vec{c}, \ldots$ are the names of the meta-variables ranging over lists of A. When there is no possibility of confusion, we may also write $[\vec{a}, b, \vec{c}]$ for the list $\vec{a} + [b] + \vec{c}$. We write $\vec{a} \approx \vec{b}$ if $\vec{a}$ is a permutation of $\vec{b}$. Observe that $\approx$ is an equivalence relation.

In some cases we will work with (finite) **sets**, which are defined to be lists, considered modulo arbitrary permutations of their elements and without regard of the number of repetitions of each element. The notation for operations on lists will be lifted to operations on sets. In particular, $\vec{a} + \vec{b}$ denotes the union of sets, and $|\vec{a}|$ stands for the cardinal of $\vec{a}$. This notation is chosen to resemble the multiset notation of existing intersection type systems. Whether we are referring to sets or lists will be clear from the context.

[2] The projection R/σ essentially stands for the work from R that is left after doing σ.

1.4 Typing

In the framework of typed calculi, a typing judgment is a statement of the meta-theory that contains the knowledge that with a given typing context, we can prove that some term has some type. For example, knowing $\Gamma \vdash t : \tau$ means that with the typing context Γ we can prove that t has type τ.

Typing contexts, or contexts for short, ranged over by $\Gamma, \Delta, \Theta, \ldots$ are (total) functions from variables to finite sets of types. We write $\mathrm{dom}\,\Gamma$ for the set of variables x such that $\Gamma(x) \neq []$. We write $\varnothing$ for the context such that $\varnothing(x) = []$ for every variable x. The notation $\Gamma + \Delta$ stands for the **sum of contexts**, defined as follows:

$$(\Gamma + \Delta)(x) \stackrel{\text{def}}{=} \Gamma(x) + \Delta(x)$$

The notation $\Gamma \oplus \Delta$ stands for the **disjoint sum of contexts**, *i.e.* it stands for $\Gamma + \Delta$ provided $\mathrm{dom}\,\Gamma \cap \mathrm{dom}\,\Delta = \varnothing$. We also write $\Gamma +_{i=1}^{n} \Delta_i$ for $\Gamma + \sum_{i=1}^{n} \Delta_i$. Moreover, $x : \mathcal{M}$ denotes the context such that $(x : \mathcal{M})(x) = \mathcal{M}$ and $\mathrm{dom}(x : \mathcal{M}) = \{x\}$.

Chapter 2

A distributive λ-calculus

In this chapter we present a *distributive λ-calculus* ($\lambda^{\#}$), and we prove some basic properties it enjoys, most importantly confluence and strong normalization.

Terms of the $\lambda^{\#}$-calculus are typing derivations of a non-idempotent intersection type system, written using proof term syntax. The underlying type system is a variant of system $\mathcal{W}$ of [BKDR14, BKV17], the main difference being that $\lambda^{\#}$ uses *labels* and a suitable invariant on terms, to ensure that the formal parameters of all functions are in 1–1 correspondence with the actual arguments that they receive.

2.1 Types

We will now present the type system we will work with, which as we said, is a variant of the system presented in [BKV17].

To make the notation easier, we will write $\tau_1 \cap ... \cap \tau_n$ as the multiset $[\tau_1, \ldots, \tau_n]$. Recall that we work in a non idempotent environment so we may have repetitions.

Definition 2.1 (Types)**.** Let $\mathscr{L} = \{\ell, \ell', \ell'', \ldots\}$ be a denumerable set of labels. The sets of **types**, ranged over by $\tau, \sigma, \rho, \ldots$, and **finite sets of types**, ranged over by $\mathcal{M}, \mathcal{N}, \mathcal{P}, \ldots$, are given mutually inductively by the following abstract syntax:

$$\tau ::= \alpha^{\ell} \mid \mathcal{M} \xrightarrow{\ell} \tau$$

$$\mathcal{M} ::= [\tau_i]_{i=1}^{n} \quad \text{for some } n \geq 0$$

In a type like α^{ℓ} and $\mathcal{M} \xrightarrow{\ell} \tau$, the label ℓ is called the **external label**.

One remark that deserves mention is that the difference with the system $\mathcal{W}$ at type-level is that we include labels. Like in system $\mathcal{W}$, note that it will not be possible for a term to have multiple types, like the name *intersection type system* would suggest. Rather, what happens is that function terms will receive a parameter that can be interpreted as having several types.

Later we will need to look closely at these types, for which purpose the following definition will be useful.

Definition 2.2. A type τ is said to **occur** in another type σ if $\tau \preceq \sigma$ holds, where $(\preceq)$ is the reflexive and transitive closure of the axioms $\tau_i \preceq [\tau_1, \dots, \tau_n] \to \sigma$ for all $i \in \{1, \dots, n\}$, and $\sigma \preceq [\tau_1, \dots, \tau_n] \to \sigma$. This is extended to say that a type τ occurs in a multiset $[\sigma_1, \dots, \sigma_n]$, defined by $\tau \preceq [\sigma_1, \dots, \sigma_n]$ if $\tau \preceq \sigma_i$ for some $i \in \{1, \dots, n\}$, and that a type τ occurs in a typing context Γ, defined by $\tau \preceq \Gamma$ if $\tau \preceq \Gamma(x)$ for some $x \in \operatorname{dom}\Gamma$.

2.2 Syntax

Generally, intersection type systems are used on the pure λ-calculus. In such environment, a term may be able to be typed: for example, in system $\mathcal{W}$, $\lambda x.xx$ can have the type $((\alpha \to \beta) \cap \alpha) \to \beta$, but the term $\Omega = (\lambda x.xx)(\lambda x.xx)$ cannot be typed.

In contrast, what we will do in this work is define a slightly different calculus, such that all well-formed terms of that calculus can be typed using the non idempotent intersection type system defined above.

The informal idea behind the definition of the terms of the calculus is that in an application, for each type in the domain of the function there will be a different argument. So if we have a term t with type $(\tau_1 \cap \tau_2) \to \tau_3$, we will apply it to a *list* of arguments, one with type τ_1 and the other with type τ_2.

Definition 2.3 (Distributive type system)**.** The set of **distributive terms**, ranged over by $(t, s, u, \dots)$ is given by the following abstract syntax:

$$t ::= x^{\tau} \mid \lambda^{\ell} x.t \mid t\,\vec{t}$$

Typing rules are defined inductively as follows.

$$\frac{}{x : [\tau] \vdash x^{\tau} : \tau}\ \texttt{var} \qquad \frac{\Gamma \oplus x : \mathcal{M} \vdash t : \sigma}{\Gamma \vdash \lambda^{\ell} x.t : \mathcal{M} \xrightarrow{\ell} \sigma}\ \to_{\mathrm{I}}$$

$$\frac{\Gamma \vdash t : [\sigma_1, \dots, \sigma_n] \xrightarrow{\ell} \tau \qquad (\Delta_i \vdash s_i : \sigma_i)_{i=1}^{n}}{\Gamma +_{i=1}^{n} \Delta_i \vdash t[s_1, \dots, s_n] : \tau}\ \to_{\mathrm{E}}$$

Moreover, we introduce a judgment of the form $[\Gamma_1, \dots, \Gamma_n] \vdash [t_1, \dots, t_n] : [\tau_1, \dots, \tau_n]$ with the following rule:

$$\frac{\Gamma_i \vdash t_i : \tau_i \text{ for all } i = 1..n}{[\Gamma_1, \dots, \Gamma_n] \vdash [t_1, \dots, t_n] : [\tau_1, \dots, \tau_n]}\ \texttt{t-multi}$$

The most noticeable feature of these terms is that in applications we don't have an argument that can be typed in many ways (as we did in system $\mathcal{W}$). Rather, we have a different term for each type that the function expects its parameter to be.

Example 2.4. *Using integer labels,*

$$\vdash \lambda^{1} x.x^{[\alpha^2, \alpha^3] \xrightarrow{4} \beta^5}[x^{\alpha^3}, x^{\alpha^2}] : [[\alpha^2, \alpha^3] \xrightarrow{4} \beta^5, \alpha^2, \alpha^3] \xrightarrow{1} \beta^5$$

is a derivable judgment.

Recién ahora, con todas estas definiciones, se entienden los ejemplos de la página xiii. Quizá hay que simplificarlos, o explicar mejor la notación.

Mostrar la derivación completa!

Note that writing all the labels can be tiresome (because writing all the labels is essentially writing a proof that the term is typed), so we will omit or simplify them when possible.

Correctness

Observe that the definition we gave has a fatal flaw: we cannot uniquely associate arguments with variables in the body of the lambdas. For example, consider the following term.

$$(\lambda^1 x.y^{[\alpha^2,\alpha^2]\xrightarrow{3}\alpha^4}[x^{\alpha^2}, x^{\alpha^2}])[a^{\alpha^2}, b^{\alpha^2}]$$

We don't know which parameter to associate which each x—which parameter goes in the first x, a or b?

To solve that problem we introduce an invariant that will ensure that problem does not manifest. We will call that invariant **correctness**.

Note that the problem is that the function in the application expects two arguments with exactly the same type. A related problem is that, in the body of the function, the variable x has the same type twice (remember that in a non idempotent context repetition matters). In fact, it is enough to ask that those two anomalies don't show up for the system to work.

We will also ask that the labels of the lambdas do not repeat which will come in handy later.

Definition 2.5 (Correct term)**.** A multiset of types $[\tau_1, \dots, \tau_n]$ is **sequential** if the external labels of τ_i and τ_j are different for all $i \neq j$. A typing context Γ is sequential if $\Gamma(x)$ is sequential for every $x \in \operatorname{dom}\Gamma$. A term t is correct if it is typable and it verifies the following three conditions:

1. *Uniquely labeled lambdas.* If $\lambda^\ell x.s$ and $\lambda^{\ell'} y.u$ are sub-terms of t at different positions, then ℓ and ℓ' must be different labels. [1]
2. *Sequential contexts.* If s is a sub-term of t and $\Gamma \vdash s : \tau$ is derivable, then Γ must be sequential.
3. *Sequential types.* If s is a sub-term of t, the judgment $\Gamma \vdash s : \tau$ is derivable, and there exists a type such that $(\mathcal{M} \xrightarrow{\ell} \sigma \preceq \Gamma)$ or $(\mathcal{M} \xrightarrow{\ell} \sigma \preceq \tau)$, then $\mathcal{M}$ must be sequential.

Essentially, correctness says that for any function that appears on the term, its parameters should be uniquely identifiable.

Example 2.6. *$x^{[\alpha^1]\xrightarrow{2}\beta^3}[x^{\alpha^1}]$ is a correct term. The example we gave above,*

$$(\lambda^1 x.y^{[\alpha^2,\alpha^2]\xrightarrow{3}\alpha^4}[x^{\alpha^2}, x^{\alpha^2}])[a^{\alpha^2}, b^{\alpha^2}]$$

is not. One problem is that the type of y is not sequential. For a last example, $\lambda^1 x.\lambda^1 y.y^{\alpha^2}$ is not a correct term since labels for lambdas are not unique.

[1] This is not strictly necessary for our current purpose, but will be useful later.

Remark 2.7. We will consider $\mathcal{T}^{\#}$ to be the set of all correct terms. In other words, we will only consider correct terms during the rest of the work.

Having defined the types and the terms of our system sheds some light on the resource management capabilities that we claimed it will enjoy: note that we can track very precisely how (*i.e.* with which type) a term will be used or a bounded variable will be evaluated. This will prove very useful to analyze the λ-calculus.

The next lemma shows that a term is uniquely typable: this means that for a given term there is only one type and typing context that satisfy the typing judgment. Moreover, all proof trees are the same.

Note that for all proof trees to be the same it is crucial that we only consider correct terms, because otherwise when we apply the rule $\to_{\mathrm{E}}$ we may have the possibility to choose between different orders for the parameters.[2]

Lemma 2.8 (Unique typing)**.** *Let t be typable,* i.e. *suppose that there exist a context Γ and a type τ such that $\Gamma \vdash t : \tau$. Furthermore, suppose that t is correct. Then there is a unique derivation that types t. In particular, if $\Gamma' \vdash t : \tau'$, then $\Gamma = \Gamma'$ and $\tau = \tau'$.*

Proof. ♠ By induction on t. □

2.3 The calculus

What we want to do know is give the operational semantics for this calculus. The idea is straightforward: in order apply a lambda abstraction to a list of arguments we just replace each occurrence of the variable bounded by the lambda with the corresponding argument.

To do that we need to define notation for the type of occurrences of a free variable. If t is typable, $\mathtt{T}_x(t)$ stands for the multiset of types of the free occurrences of x in t. If $t_1, \dots, t_n$ are typable, $\mathtt{T}([t_1, \dots, t_n])$ stands for the multiset of types of $t_1, \dots, t_n$. For example, $\mathtt{T}_x(x^{[\alpha^1] \xrightarrow{2} \beta^3}[x^{\alpha^1}]) = \mathtt{T}([y^{\alpha^1}, z^{[\alpha^1] \xrightarrow{2} \beta^3}]) = [[\alpha^1] \xrightarrow{2} \beta^3, \alpha^1]$. To perform a substitution $t\{x := [s_1, \dots, s_n]\}$ we will require that $\mathtt{T}_x(t) = \mathtt{T}([s_1, \dots, s_n])$.

Let us define exactly what we mean by *substitution*.

Definition 2.9 (Substitution)**.** Let t and $s_1, \dots, s_n$ be correct terms such that $\mathtt{T}_x(t) = \mathtt{T}([s_1, \dots, s_n])$. The capture-avoiding substitution of x in t by $\vec{s} = [s_1, \dots, s_n]$ is denoted by $t\{x := \vec{s}\}$ and defined as follows:

$$
\begin{array}{rcll}
x^{\tau}\{x := [s]\} & \stackrel{\text{def}}{=} & s & \\
y^{\tau}\{x := []\} & \stackrel{\text{def}}{=} & y^{\tau} & \text{if } x \neq y \\
(\lambda^{\ell} y.u)\{x := \vec{s}\} & \stackrel{\text{def}}{=} & \lambda^{\ell} y.u\{x := \vec{s}\} & \text{if } x \neq y \text{ and } y \notin \mathsf{fv}(\vec{s}) \\
u_0[u_j]_{j=1}^{m}\{x := \vec{s}\} & \stackrel{\text{def}}{=} & u_0\{x := \vec{s}_0\}[u_j\{x := \vec{s}_j\}]_{j=1}^{m} &
\end{array}
$$

In the last case, $(\vec{s}_0, \dots, \vec{s}_m)$ is a partition of $\vec{s}$ such that $\mathtt{T}_x(u_j) = \mathtt{T}(\vec{s}_j)$ for all $j \in 0, \dots, m$.

[2] This implies that a weaker uniqueness result holds for incorrect terms: proof trees are equal modulo permutations when the rule $\to_{\mathrm{E}}$ is applied, but as we don't care about incorrect terms we don't need to consider this.

For example,

$$(x^{[\alpha^1]\overset{2}{\to}\beta^3}[x^{\alpha^1}])\{x := [y^{[\alpha^1]\overset{2}{\to}\beta^3}, z^{\alpha^1}]\} = y^{[\alpha^1]\overset{2}{\to}\beta^3} z^{\alpha^1}$$

and

$$(x^{[\alpha^1]\overset{2}{\to}\beta^3}[x^{\alpha^1}])\{x := [y^{\alpha^1}, z^{[\alpha^1]\overset{2}{\to}\beta^3}]\} = z^{[\alpha^1]\overset{2}{\to}\beta^3} y^{\alpha^1}.$$

Remark 2.10. Substitution is *type-directed*: arguments $[s_1, \ldots, s_n]$ are propagated throughout the term so that s_i reaches the free occurrence of x that has the same type as s_i. There exists one such occurrence for each $i \in \{1, \ldots, n\}$ because $\mathtt{T}_x(t) = \mathtt{T}([s_1, \ldots, s_n])$. Moreover, the fact that t is correct ensures that such occurrence is unique, since $\mathtt{T}_x(t)$ is sequential. The following lemma formalizes what we just said.

Lemma 2.11 (Substitution is well-defined)**.** *If $\Gamma \oplus x : \vec{\sigma} \vdash t : \tau$ and $\vec{\Delta} \vdash \vec{s} : \vec{\sigma}$ are derivable, then $\Gamma + \vec{\Delta} \vdash t\{x := \vec{s}\} : \tau$ is derivable.*

Proof. By induction on t, straightforward using the ideas in the last remark. □

Substitution as we presented it can be hard to work with in some environments, as we have to track how substitution terms get distributed over the term they are being substituted in.

The following alternative definition 🚀 deals with this by taking advantage of the fact that substitution is type-directed: it takes all terms all the way down and does not split them in the application case, but rather "picks" the correct term to substitute in the base case.

Definition 2.12 (Alternative definition of substitution)**.** Let $\Gamma, x : [\tau_1, \ldots, \tau_n] \vdash t : \sigma$ and let $\Delta_i \vdash s_i : \rho_i$ for each $i \in \{1, \ldots, m\}$ in such a way that $\mathtt{T}_x(t) \subseteq \mathtt{T}([s_1, \ldots, s_m])$ and $[\sigma_1, \ldots, \sigma_n]$ is sequential. Let us write $\vec{s}$ for $[s_1, \ldots, s_m]$. Then an alternative definition for substitution $t\{\!\{x := \vec{s}\}\!\}$ may be defined as follows:

$$\begin{array}{rcll}
x^\tau\{\!\{x := \vec{s}\}\!\} & \overset{\text{def}}{=} & s_i & \text{where } i \text{ is the unique index such that } \mathtt{T}(s_i) = \tau \\
y^\tau\{\!\{x := \vec{s}\}\!\} & \overset{\text{def}}{=} & y^\tau & \text{if } x \neq y \\
(\lambda^\ell y.t)\{\!\{x := \vec{s}\}\!\} & \overset{\text{def}}{=} & \lambda^\ell y.t\{\!\{x := \vec{s}\}\!\} & \text{if } x \neq y \text{ and there is no capture} \\
(t[u_i]_{i=1}^k)\{\!\{x := \vec{s}\}\!\} & \overset{\text{def}}{=} & t\{\!\{x := \vec{s}\}\!\}[u_i\{\!\{x := \vec{s}\}\!\}]_{i=1}^k. &
\end{array}$$

Moreover, $[t_1, \ldots, t_n]\{\!\{x := \vec{s}\}\!\}$ stands for $[t_1\{\!\{x := \vec{s}\}\!\}, \ldots, t_n\{\!\{x := \vec{s}\}\!\}]$, whenever each substitution $t_i\{\!\{x := \vec{s}\}\!\}$ is well-defined.

Lemma 2.13. *$t\{x := \vec{s}\} = t\{\!\{x := \vec{s}'\}\!\}$ whenever $\mathtt{T}_x(t) = \mathtt{T}(\vec{s})$ and $\vec{s}$ is a sub-list of $\vec{s}'$.*

Proof. By induction on t.

1. **Variable (same),** $t = x^\tau$**.** First $x^\tau\{x := [s]\} = s$. Also,$x^\tau\{\!\{x := \vec{s}'\}\!\} = s$, because s is in $\vec{s}'$ and it must be the only one for which the external label is the external label of τ.
2. **Variable (different),** $t = y$**.** On the left hand side, $y^\sigma\{x := [\,]\} = y^\sigma$. On the right hand side, $y^\sigma\{\!\{x := \vec{s}'\}\!\} = y^\sigma$.

3. **Abstraction,** $t = \lambda^\ell y.u$**.** On the left, $(\lambda^\ell y.u)\{x := \vec{s}\} = \lambda^\ell y.u\{x := \vec{s}\}$. On the right $(\lambda^\ell y.u)\{\!\{x := \vec{s}'\}\!\} = \lambda^\ell y.u\{\!\{x := \vec{s}\}\!\}'$. The right-hand side of both equations are the same by inductive hypothesis.
4. **Application,** $t = r[u_1, \ldots, u_n]$**.** On the left side we have that $(r[u_i]_{i=1}^n)\{x := \vec{s}\} = r\{x := \vec{s}_0\}[u_i\{x := \vec{s}_i\}]_{i=1}^n$, where $\vec{s}_0 +_{i=1}^n \vec{s}_i$ is a permutation of $\vec{s}$, $\mathtt{T}_x(t) = \mathtt{T}(\vec{s}_0)$ and $\mathtt{T}_x(u_i) = \mathtt{T}(\vec{s}_i)$ for all $i = \{1, ..., n\}$. On the other hand, $((r[u_i]_{i=1}^n))\{\!\{x := \vec{s}'\}\!\} = r\{\!\{x := \vec{s}'\}\!\}[u_i\{\!\{x := \vec{s}'\}\!\}]_{i=1}^n$.
Note that for every $i \in \{0, ..., n\}$, $\vec{s}_i$ is a sub-list of $\vec{s}$. Then, by inductive hypothesis, $r\{x := \vec{s}_0\} = r\{\!\{x := \vec{s}'\}\!\}$ and $u_i\{x := \vec{s}_i\} = u_i\{\!\{x := \vec{s}'\}\!\}$, which is what we wanted.

□

Lemma 2.14 (Substitution lemma for the alternative notion of substitution)**.** *The following variant of the substitution lemma holds when* $x \notin \mathsf{fv}(\vec{u})$*:*

$$t\{\!\{x := \vec{s}\}\!\}\{\!\{y := \vec{u}\}\!\} = t\{\!\{y := \vec{u}\}\!\}\{\!\{x := \vec{s}\{\!\{y := \vec{u}\}\!\}\}\!\}$$

This equation is intended to mean, in particular, that one side is well-defined if and only if the other side is well-defined.

Proof. Straightforward by induction on t. □

Remark 2.15. $\mathtt{C}\langle t\rangle\{\!\{x := \vec{s}\}\!\} = \mathtt{C}\{\!\{x := \vec{s}\}\!\}\langle t\{\!\{x := \vec{s}\}\!\}\rangle$.

Notación no introducida aún

Definition 2.16 (The $\lambda^\#$-calculus)**.** The $\lambda^\#$**-calculus** (distributive λ-calculus) is given by the set of correct typable terms $\mathcal{T}^\#$. For each label $\ell \in \mathscr{L}$, we define a reduction relation $\xrightarrow{\ell}_\# \subseteq \mathcal{T}^\# \times \mathcal{T}^\#$ as follows:

$$\mathtt{C}\langle(\lambda^\ell x.t)\vec{s}\rangle \xrightarrow{\ell}_\# \mathtt{C}\langle t\{x := \vec{s}\}\rangle$$

where $\mathtt{C}$ stands for a **context**. The binary relation $\to_\#$ is the union of all the $\xrightarrow{\ell}_\#$:

$$\to_\# \stackrel{\text{def}}{=} \bigcup_{\ell \in \mathscr{L}} \xrightarrow{\ell}_\#$$

We sometimes drop the subscript for $\to_\#$, writing just $\to$, when clear from the context. The set of contexts is given by the grammar:

$$\mathtt{C} ::= \Box \mid \lambda^\ell x.\mathtt{C} \mid \mathtt{C}\,\vec{t} \mid t[s_1, \ldots, s_{i-1}, \mathtt{C}, s_{i+1}, \ldots, s_n]$$

Contexts can be thought as terms with a single free occurrence of a distinguished variable $\Box$. The notation $\mathtt{C}\langle t\rangle$ stands for the capturing substitution of the occurrence of $\Box$ in $\mathtt{C}$ by t.

In general an n-hole context is a term $\mathtt{C}$ with exactly $n \geq 0$ free occurrences of the distinguished variable $\Box$. If $\mathtt{C}$ is an n-hole context, $\mathtt{C}\langle t_1, \ldots, t_n\rangle$ stands for the term that results from performing the substitution of the i-th occurrence of $\Box_i$ (from left to right) by t_i. If $\mathtt{C}$ is an n-hole context for some n, we say that it is a many-hole context.

La def. no permite + de 1 □

2.4 Basic properties

The first lemma talks about the shape of the typing context of a given typing judgment. Specifically, that the typing context will contain $x : \tau$ for each free variable x^τ that occurs in the term, and nothing else.

This lemma will be very useful to prove a lot of upcoming results.

Lemma 2.17 (Linearity). *Let $t \in \mathcal{T}^\#$ be a correct term, and let $\Gamma \vdash t : \tau$ be the (unique) type derivation for t. Let x be any variable, and consider the $n \geq 0$ free occurrences of the variable x in the term t, more precisely, write t as $t = \hat{C}\langle x^{\tau_1}, \ldots, x^{\tau_n}\rangle$, where $\hat{C}$ is a context with n-holes such that $x \notin \mathsf{fv}(\hat{C})$.*

Proof. ♠ By induction on t. □

Remark 2.18. Given that variables are labeled with their types, it is more or less easy to obtain the type of a given (correct and typable) term. Moreover, Linearity (Lemma 2.17) shows that it is easy to obtain the context of the typing judgment.

In summary, given a term t that is correct and typable it is straightforward to obtain its typing judgment $\Gamma \vdash t : \tau$. It is also straightforward to find out whether the term is typable or not—the typability will manifest itself while we try to find τ.

The following properties show that the rewrite rule $\to_\#$ is well-defined.

Lemma 2.19 ($\mathcal{T}^\#$ is closed under $\to_\#$). *Let $t \in \mathcal{T}^\#$ such that $t \to_\# t'$. Then $t' \in \mathcal{T}^\#$.*

Proof. ♠ Let $t = \mathtt{C}\langle(\lambda^\ell x.s)\vec{u}\rangle$. By induction on $\mathtt{C}$. □

Lemma 2.20 (Subject reduction). *If $\Gamma \vdash \mathtt{C}\langle(\lambda^\ell x.t)\vec{s}\rangle : \tau$ then $\Gamma \vdash \mathtt{C}\langle t\{x := \vec{s}\}\rangle : \tau$. Moreover, correctness is preserved.*

Proof. ♠ By induction on $\mathtt{C}$. □

These two lemmas are very important because they show that $\to_\#$ behaves well, in particular that it preserves correctness of terms and their types. That is, if $t \to t'$ and t is correct and well-typed, we know that t' is correct, well-typed and has the same type as t.

Termination

The $\lambda^\#$-calculus happens to be strongly normalizing. This is because substitution is *linear*, *i.e.* the term $t\{x := [s_1, \ldots, s_n]\}$ uses s_i exactly once for all $i \in \{1, \ldots, n\}$, hence $\to_\#$ reduces the number of lambdas of a term in exactly one.

Proposition 2.21 (Termination). *There is no infinite reduction $t_0 \to_\# t_1 \to_\# t_2 \to_\# \ldots$.*

Proof. It suffices to show that there is a function $d : \mathcal{T}^\# \to \mathbb{N}_0$ compatible with $\to_\#$, *i.e.* such that $t \to_\# s$ implies $d(t) > d(s)$. In particular, we will show that taking $d(t)$ to be the number of λs in t works. More precisely:

$$d(x^\tau) \stackrel{\text{def}}{=} 0$$

$$d(\lambda^\ell x.t) \stackrel{\text{def}}{=} 1 + d(t)$$

$$d(t[s_1, \ldots, s_n]) \stackrel{\text{def}}{=} d(t) + \sum_{i=1}^{n} d(s_i)$$

This definition may be extended to contexts, by taking $d(\Box) \stackrel{\text{def}}{=} 0$. It is straightforward to show, by induction on $\mathtt{C}$, that $d(\mathtt{C}\langle t\rangle) = d(\mathtt{C}) + d(t)$.

Now, to prove the proposition, suppose that

$$\mathtt{C}\langle(\lambda^{\ell} x.t)\vec{s}\rangle \to_{\#} \mathtt{C}\langle t\{x := \vec{s}\}\rangle$$

We would like to see that $d(\mathtt{C}\langle(\lambda^{\ell} x.t)\vec{s}\rangle) > d(\mathtt{C}\langle t\{x := \vec{s}\}\rangle)$. But

$$\begin{array}{rcl} d(\mathtt{C}\langle(\lambda^{\ell} x.t)\rangle) & = & d(\mathtt{C}) + d((\lambda^{\ell} x.t)\vec{s}) \\ & = & d(\mathtt{C}) + d(\lambda^{\ell} x.t) + \sum_{i=1}^{n} d(s_i) \\ & = & d(\mathtt{C}) + 1 + d(t) + \sum_{i=1}^{n} d(s_i) \end{array}$$

Moreover

$$d(\mathtt{C}\langle t\{x := \vec{s}\}\rangle) = d(\mathtt{C}) + d(t\{x := \vec{s}\})$$

So it suffices to show that $1 + d(t) + \sum_{i=1}^{n} d(s_i) > d(t\{x := \vec{s}\})$. As a matter of fact, a stronger proposition holds: $d(t) + \sum_{i=1}^{n} d(s_i) = d(t\{x := \vec{s}\})$. ♠ We can prove this by induction on t. □

Corollary 2.22 (Bound for the length of derivations)**.** *Let $t' \in \mathcal{T}^{\#}$ be a correct term. Then there is a bound for the length of derivations starting at t'.*

Proof. This is an immediate consequence of the fact that the distributive lambda-calculus is strongly normalizing (Proposition 2.21) and finitely branching, by König's lemma. □

Remark 2.23. A possible bound—albeit not very good—is the number of lambdas of the term. This fact stems from the proof of Proposition 2.21.

Confluence

As we stated in the introduction, the goal of labeling terms and types is that we obtain a confluent calculus. Confluence means that every two reduction sequences from a term can be extended to a common reduct. Indeed, the $\lambda^{\#}$-calculus is confluent, and it is the purpose of this section to prove so.

First, we need to prove an adaptation of the Substitution Lemma for our calculus, which is a key tool to prove properties about coinitial steps. The substitution lemma for the pure lambda calculus [Bar84, Lemma 2.1.16] states that, provided that $x \neq y$ and $x \notin \mathsf{fv}(u)$, then $t\{x := s\}\{y := u\} = t\{y := u\}\{x := s\{y := u\}\}$

In our case variables get substituted by a *list* of terms, so we need to adapt it. Particularly, the list of terms that will take the place of u needs to be divided up in several lists: one for the corresponding ys in t, and the rest for the ys in each of the elements of s, which recall that now will be a list.

Notation 2.24. We extend the substitution operator to work on lists, defining

$$[t_i]_{i=1}^{n}\{x := \vec{s}\} \stackrel{\text{def}}{=} [t_i\{x := \vec{s}_i\}]_{i=1}^{n}$$

where $(\vec{s}_1, \ldots, \vec{s}_n)$ is a partition of $\vec{s}$ such that $\mathtt{T}_x(t_i) = \mathtt{T}(\vec{s}_i)$ for all $i \in \{1, \ldots, n\}$.

Lemma 2.25 (Substitution Lemma)**.** *Let $x \neq y$ and $x \notin \mathsf{fv}(\vec{u})$. If $(\vec{u}_1, \vec{u}_2)$ is a partition of $\vec{u}$ then*

$$t\{x := \vec{s}\}\{y := \vec{u}\} = t\{y := \vec{u}_1\}\{x := \vec{s}\{y := \vec{u}_2\}\}$$

provided that both sides of the equation are defined. Note: *there exists a list $\vec{u}$ that makes the left-hand side defined if and only if there exist lists $\vec{u}_1, \vec{u}_2$ that make the right-hand side defined.*
Proof. ♠ By induction on t. □

Example 2.26. *For example, consider the term*

$$t = (\lambda^{\ell} x.z^{[\alpha]\to[\beta]\to\delta}[x^{\alpha}][x^{\beta}])[z^{\beta}, y^{\alpha}].$$

We can perform the following substitution (where w, a, b are all variables):

$$t\{z := [w^{[\alpha]\to[\beta]\to\delta}, y^{\beta}]\}\{y := [a^{\beta}, b^{\alpha}]\} = (\lambda^{\ell} x.w^{[\alpha]\to[\beta]\to\delta}[x^{\alpha}][x^{\beta}])[a^{\beta}, b^{\alpha}].$$

Note that, as we replaced one z by a y, if we wanted to invert the order of the substitutions we would need to separate the list $[a^{\beta}, b^{\alpha}]$ in two:

$$t\{y := [b^{\alpha}]\}\{z := [w^{[\alpha]\to[\beta]\to\delta}, y^{\beta}]\{y := a^{\beta}\}\} = (\lambda^{\ell} x.w^{[\alpha]\to[\beta]\to\delta}[x^{\alpha}][x^{\beta}])[a^{\beta}, b^{\alpha}].$$

Although are going to prove that the $\lambda^{\#}$-calculus is confluent, actually a strictly stronger property holds: strong permutation. This property says that if we "open" a diagram with two (different) steps, we can "close" it with two steps too; and it also gives us information about which are those closing steps, via their labels. The fact that this stronger property holds gives us the chance to develop a functorial residual theory (*cf.* Full Stability, Lemma 3.20), as we will learn in the next chapter.

Proposition 2.27 (Strong Permutation). *If $t_0 \xrightarrow{\ell_1}_{\#} t_1$ and $t_0 \xrightarrow{\ell_2}_{\#} t_2$ are different steps, then there exists a term $t_3 \in \mathcal{T}^{\#}$ such that $t_1 \xrightarrow{\ell_2}_{\#} t_3$ and $t_2 \xrightarrow{\ell_1}_{\#} t_3$. Diagrammatically,*

$$\begin{array}{ccc} t_0 & \xrightarrow{\ell_1} & t_1 \\ \downarrow{\scriptstyle \ell_2} & & \downarrow{\scriptstyle \ell_2} \\ t_2 & \xrightarrow{\ell_1} & t_3. \end{array}$$

Proof. Let $R : t_0 \xrightarrow{\ell}_{\#} t_1$ and $S : t_0 \xrightarrow{\ell'}_{\#} t_2$ be steps going out from t_0, and let us show that the peak may be closed. The step R is of the form:

$$R : t_0 = \mathtt{C}\langle(\lambda^{\ell} x.t)\vec{s}\rangle \xrightarrow{\ell}_{\#} \mathtt{C}\langle t\{x := \vec{s}\}\rangle = t_1$$

We proceed by induction on $\mathtt{C}$, and within each case we separate in different cases depending on where S is located (which recall that is different than R by hypothesis). ♠ □

The Strong Permutation property may also be called *sub-commutativity* or $WCR^{\leq 1}$ in some contexts. [3] As a consequence of strong permutation, reduction is sub-commutative,

[3] In reality, sub-commutativity and $WCR^{\leq 1}$ are slightly different than strong permutation, but equivalent: as they don't ask for the steps to be different, but allow the closing steps to be at most one (instead of exactly one).

i.e. $(\leftarrow_{\#} \circ \rightarrow_{\#}) \subseteq (\rightarrow_{\#}^{=} \circ \leftarrow_{\#}^{=})$ where $\leftarrow_{\#}$ denotes $(\rightarrow_{\#})^{-1}$ and $R^{=}$ denotes the reflexive closure of R.

Moreover, it is well-known that sub-commutativity implies confluence, *i.e.* $(\leftarrow_{\#}^{*} \circ \rightarrow_{\#}^{*}) \subseteq (\rightarrow_{\#}^{*} \circ \leftarrow_{\#}^{*})$ (*cf.* [Ter03, Proposition 1.1.10]). Note that the inverse does not hold, hence confluence is weaker than strong permutation, as we previously stated.

Corollary 2.28 (Confluence)**.** *$\lambda^{\#}$ is confluent,* i.e.*, if $t_0 \twoheadrightarrow_{\#} t_1$ and $t_0 \twoheadrightarrow_{\#} t_2$, then there exists a term t_3 such that $t_1 \twoheadrightarrow_{\#} t_3$ and $t_2 \twoheadrightarrow_{\#} t_3$. Diagrammatically,*

$$\begin{array}{ccc} t_0 & \twoheadrightarrow & t_1 \\ \downarrow & & \vdots \\ t_2 & \dashrightarrow & t_3. \end{array}$$

Chapter 3

Residual theory

In this section we will develop the theory of residuals of the $\lambda^{\#}$-calculus. We do this as it will help us to prove that derivation spaces of $\lambda^{\#}$ terms have a very simple structure. This together with the fact that the $\lambda^{\#}$-calculus can simulate the λ-calculus in a strong sense (as we will learn in the next chapter) will make the $\lambda^{\#}$-calculus a very useful tool to analyze the pure λ-calculus.

The basic concepts of general rewriting theory were outlined in the preliminaries (Section 1.2), and we will now concentrate in the specifics of our calculus.

Informally, the residual of a step after another is what is left of a step after executing another one; it is a set of steps.

Example 3.1. *If we consider the term of the pure lambda calculus:*

$$(\lambda x.y)((\lambda z.z)w)$$

Then there are two steps we can perform (we shall call them R and S).

$$R : (\lambda x.y)((\lambda z.z)w) \to y$$
$$S : (\lambda x.y)((\lambda z.z)w) \to (\lambda x.y)w$$

The residual of R after S is the step $(\lambda x.y)w \to w$. On the other hand, the residual of S after R is the empty set: S was erased by R.

It is interesting to develop a theory of residuals for $\lambda^{\#}$ because residuals allow us to trace a redex through the reduction of a term. Given that the purpose of $\lambda^{\#}$ is to be able to track how resources are used in λ-terms, it is crucial to be able to learn how these resources interact during a reduction.

As it turns out, the theory of residuals of $\lambda^{\#}$ will prove to be powerful enough to represent meaningful information, but simple enough to have a comprehensible structure.

Definition 3.2. If $R : t \xrightarrow{\ell}_{\#} t'$ is a step in the distributive lambda calculus, then:

$$\begin{array}{rcll} \mathsf{src}(R) & \stackrel{\text{def}}{=} & t & \text{is the } \textbf{source} \text{ of } R \\ \mathsf{tgt}(R) & \stackrel{\text{def}}{=} & t' & \text{is the } \textbf{target} \text{ of } R \\ \mathsf{name}(R) & \stackrel{\text{def}}{=} & \ell & \text{is the } \textbf{name} \text{ of } R \end{array}$$

Two steps R and S are **coinitial** if $\mathsf{src}(R) = \mathsf{src}(S)$ and **cofinal** if $\mathsf{tgt}(R) = \mathsf{tgt}(S)$.

Definition 3.3 (Residuals in the distributive lambda-calculus)**.** Given coinitial steps R, S, the set R/S of **residuals of** R **after** S is defined as follows:

$$R/S = \{R' \mid \mathsf{src}(R') = \mathsf{tgt}(S) \text{ and } \mathsf{name}(R) = \mathsf{name}(R')\}$$

Remark 3.4. Recall that the name of a step is the label that decorates the lambda reduced by the step, and that in correct terms all lambdas have pairwise distinct labels. Given that our calculus has no duplication or erasure (as per the following lemma), names of steps will be useful to name reductions.

Lemma 3.5 (Cardinality of the set of residuals)**.**

$$\#(R/S) = \begin{cases} 0 & \text{if } R = S \\ 1 & \text{otherwise} \end{cases}$$

Proof. Recall that, by definition, lambdas in a correct term have pairwise distinct labels. Consider first the case when $R = S$. Then $R = S : \mathtt{C}\langle(\lambda^{\ell} x.t)\vec{s}\rangle \xrightarrow{\ell}_{\#} \mathtt{C}\langle t\{x := \vec{s}\}\rangle$. There is only one lambda decorated with ℓ in the source, so there are no lambdas decorated with ℓ in the target. Hence $R/S = \varnothing$.

On the other hand if, $R \neq S$, by the Strong Permutation property (Proposition 2.27) there exists a step $R' \in R/S$ with the same name as R. There are no other lambdas decorated with ℓ in the target. Hence $R/S = \{R'\}$. □

3.1 Orthogonality of $\lambda^{\#}$

Recall from the preliminaries that some abstract rewriting systems have the property of being *orthogonal*. Being orthogonal entails a myriad of properties and results that make working with orthogonal rewrite systems very pleasant. Informally, in an orthogonal rewrite system residuals behave and have the properties that one would expect, some of which are summarized in the table that follows.

$\epsilon\,\rho = \rho$	$\rho \sqsubseteq \sigma \stackrel{\text{def}}{\iff} \rho/\sigma = \epsilon$	$\rho \sqsubseteq \sigma \implies \rho/\tau \sqsubseteq \sigma/\tau$
$\rho\,\epsilon = \rho$	$\rho \equiv \sigma \stackrel{\text{def}}{\iff} \rho \sqsubseteq \sigma \wedge \sigma \sqsubseteq \rho$	$\rho \sqsubseteq \sigma \iff \tau\rho \sqsubseteq \tau\sigma$
$\epsilon/\rho = \epsilon$	$\rho \sqcup \sigma \stackrel{\text{def}}{=} \rho(\sigma/\rho)$	$\rho \sqcup \sigma \equiv \sigma \sqcup \rho$
$\rho/\epsilon = \rho$	$\rho \equiv \sigma \implies \tau/\rho = \tau/\sigma$	$(\rho \sqcup \sigma) \sqcup \tau = \rho \sqcup (\sigma \sqcup \tau)$
$\rho/\sigma\tau = (\rho/\sigma)/\tau$	$\rho \sqsubseteq \sigma \iff \exists\tau.\, \rho\tau \equiv \sigma$	$\rho \sqsubseteq \rho \sqcup \sigma$
$\rho\sigma/\tau = (\rho/\tau)(\sigma/(\tau/\rho))$	$\rho \sqsubseteq \sigma \iff \rho \sqcup \sigma \equiv \sigma$	$(\rho \sqcup \sigma)/\tau = (\rho/\tau) \sqcup (\sigma/\tau)$
$\rho/\rho = \epsilon$		

As we stated in the preliminaries, one must check four axioms in order to prove that a given system (in this case $\lambda^{\#}$) is orthogonal: Autoerasure, Finite Residuals, Finite Developments, and Semantic Orthogonality.

Let's see why the axiom called Finite Developments is interesting. As stated in [Mel96], the *Axiome FD*, or finite developments axiom, asks that for every set of coinitial steps $\mathcal{M}$, then all developments of $\mathcal{M}$ are finite.

This axiom is important because suppose we have two coinitial steps R and S, such that $R : t \to s$. Then S/R is a set of steps with the same source (s). The idea is that if we had t and wanted to "execute" both R and S, if we didn't have finite developments, then in particular we don't have finite complete developments, so a reduction that tries to execute what's left of S after R may not finish.

The $\lambda^{\#}$-calculus not only enjoys of finite developments, but we can give the exact length a complete development of a set of steps will have.

Lemma 3.6 (Finite developments)**.** *Let $\mathcal{M}$ be a set of coinitial steps. Then the length of every complete development of $\mathcal{M}$ is precisely the cardinality of $\mathcal{M}$. In particular, developments are finite.*

Proof. By induction on the cardinality of $\mathcal{M}$. If $\mathcal{M} = \varnothing$, the only complete development of $\mathcal{M}$ is ϵ and we are done. Otherwise, if ρ is a complete development of $\mathcal{M}$, it is a non-empty derivation, *i.e.* $\rho = R\sigma$ where $R \in \mathcal{M}$ and such that σ is a complete development of $\mathcal{M}/R$. Since residuals of distinct redexes have distinct names (and hence they are distinct) we have that $\mathcal{M}/R = \uplus_{S\in\mathcal{M}}(S/R)$, where $\uplus$ denotes the disjoint union of sets. Moreover, $\#(S/R) = 1$ if and only if $R \neq S$ by Lemma 3.5, so:

$$\#(\mathcal{M}/R) = \Sigma_{S\in\mathcal{M}}\#(S/R) = \#(\mathcal{M} \setminus \{R\}) = \#(\mathcal{M}) - 1$$

Hence by *i.h.* the length of σ is $\#(\mathcal{M}) - 1$ and we conclude. □

Proposition 3.7. *The distributive lambda-calculus is an Orthogonal Axiomatic Rewrite System in the sense of Melliès.*

Proof. There are four axioms to check:

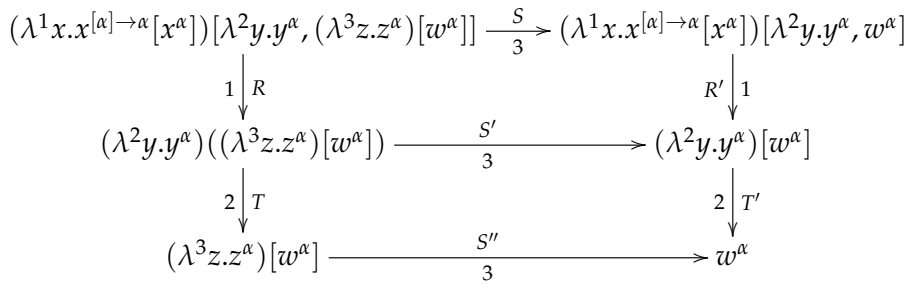
1. **Autoerasure.** Immediate from the cardinality of residuals lemma (Lemma 3.5).
2. **Finite Residuals.** Immediate from the cardinality of residuals lemma (Lemma 3.5).
3. **Finite Developments.** Proved in the Finite Developments lemma (Lemma 3.6).
4. **Semantic Orthogonality.** A consequence of the Strong Permutation property (Proposition 2.27).

□

Example 3.8. *Next we present an example that will be useful to illustrate some of the upcoming propositions. Consider the following term:*

$$(\lambda^{\ell_1} x.x^{[\alpha]\to\alpha}[x^{\alpha}])[\lambda^{\ell_2} y.y^{\alpha}, (\lambda^{\ell_3} z.z^{\alpha})[w^{\alpha}]].$$

What follows is its derivation space, i.e.*, all possible derivations that have the previous term as a source.*



Some facts about these derivations:

- R creates T: we cannot perform T until we performed R. We will expand on the topic of creation later.
- $RS' \equiv SR'$. Remember that two derivations ρ, σ are permutation equivalent if $\rho \sqsubseteq \sigma \sqsubseteq \rho$, *i.e.* if they perform the same amount of work.
- $S/R = \{R''\}$ and $S/RT = \{S''\}$. Remember that in $\lambda^\#$, residuals are either empty or singletons, so we will often skip the parentheses.
- $\mathsf{name}(T) = \mathsf{name}(T') = 3$.

3.2 Names and labels

The fact that in $\lambda^\#$ names of steps are given by labels of lambdas, which by correctness are pairwise different, and that the calculus has no deletion nor duplication will make names of steps suitable to name derivations (*i.e.* series of steps).

Furthermore, giving names to derivations will make them very easy to analyze, as we will see in this section.

Definition 3.9 (Set of names of a derivation)**.** If ρ is a derivation, its set of names is

$$\mathsf{names}(\rho) \stackrel{\text{def}}{=} \{\mathsf{name}(R) \mid \exists \rho_1, \rho_2.\ \rho = \rho_1 R \rho_2\}$$

A more precise way to say that our calculus has no duplication nor deletion is the following lemma.

Lemma 3.10 (Permanence)**.** *If $R \notin \rho$ then R/ρ is a singleton.*

Proof. By induction on ρ. The base case is trivial, so let $\rho = S\sigma$. Note that $R \neq S$, so by Lemma 3.5 we have that $R/S = S'$, where $S' \notin \sigma$. So $R/S\sigma = S'/\sigma$ is a singleton by *i.h.*. □

Belonging

There are several notions that can define the fact that a step R is performed in a derivation ρ (where R and ρ are coinitial).

One of this notions is the one that says that whatever R does, is also done by ρ, *i.e.*, $R/\rho = \varnothing$: there is no more R-related work to do. This is usually written as $R \sqsubseteq \rho$. This notion can be easily extended to derivations: $\sigma \sqsubseteq \rho$ if $\sigma/\rho = \epsilon$.

Another (weaker) notion asks that *some* of the work that R does be done in ρ. This is the notion of belonging. A step R **belongs to** a coinitial derivation ρ, written $R \in \rho$, if and only if some residual of R is contracted along ρ. More precisely, $R \in \rho$ if there exist ρ_1, R', ρ_2 such that $\rho = \rho_1 R' \rho_2$ and $R' \in R/\rho_1$. We write $R \notin \rho$ if it is not the case that $R \in \rho$.

As it turns out, these two notions are equivalent in our calculus, because there is no deletion nor duplication of redexes.

Lemma 3.11 (Characterization of belonging)**.** *Let R be a step and ρ a coinitial derivation in the distributive lambda-calculus. Then the following are equivalent:*

1. $R \in \rho$,
2. $R \sqsubseteq \rho$,
3. $\mathsf{name}(R) \in \mathsf{names}(\rho)$.

Note: *the hypothesis that R and ρ are coinitial is crucial. In particular,* (1) *and* (2) *by definition only hold when R and ρ are coinitial, while* (3) *might hold even if R and ρ are not coinitial.*

Proof. $(1 \Rightarrow 2)$ Let $\rho = \rho_1 S \sigma_2$ where S is a residual of R. Suppose moreover, without loss of generality, that ρ_1 is minimal, *i.e.* that $R \notin \rho_1$. By Permanence (Lemma 3.10) R/ρ_1 is a singleton, so $R/\rho_1 = S$. This means that $R/\rho_1 S \rho_2 = \varnothing$, so indeed $R \sqsubseteq \rho_1 S \rho_2$.

$(2 \Rightarrow 3)$ By induction on ρ. If ρ is empty, the implication is vacuously true, so let $\rho = S\sigma$ and consider two sub-cases, depending on whether $R = S$. If $R = S$, then indeed the first step of $\rho = R\sigma$ has the same name as R. On the other hand, if $R \neq S$, then by Lemma 3.5 we have that $R/S = R'$, where $\mathsf{name}(R) = \mathsf{name}(R')$. Note that $R \sqsubseteq S\sigma$ so $R/S = R' \sqsubseteq \sigma$. By applying the *i.h.* we obtain that there must be a step in σ whose name is $\mathsf{name}(R) = \mathsf{name}(R')$, and we are done.

$(3 \Rightarrow 1)$ By induction on ρ. If ρ is empty, the implication is vacuously true, so let $\rho = S\sigma$ and consider two sub-cases, depending on whether $\mathsf{name}(R) = \mathsf{name}(S)$. If $\mathsf{name}(R) = \mathsf{name}(S)$ then R and S must be the same step, since terms are correct, which means that labels decorating lambdas are pairwise distinct. Hence $R \in \rho = R\sigma$. On the other hand, if $\mathsf{name}(R) \neq \mathsf{name}(S)$, then $R \neq S$ so by Lemma 3.5 we have that $R/S = R'$, where $\mathsf{name}(R) = \mathsf{name}(R')$. By hypothesis, there is a step in the derivation $\rho = S\sigma$ whose name is $\mathsf{name}(R)$, and it is not S, so there must be at least one step in the derivation σ whose name is is $\mathsf{name}(R) = \mathsf{name}(R')$. By *i.h.* $R' \in \sigma$ and then, since R' is a residual of R, we conclude that $R \in S\sigma$, as required. □

Name of a reduction

In this subsection we will see why the naming scheme we proposed for steps and reductions is useful. Recall that the names of a reduction are the set of names of the steps that form it. What follows is a list of different results that characterize properties of reductions in terms of its name.

It is immediate to note that when composing two derivations ρ, σ, the set of names of $\rho\sigma$ results from the union of the names of ρ and σ:

Remark 3.12. $\mathsf{names}(\rho\sigma) = \mathsf{names}(\rho) \cup \mathsf{names}(\sigma)$

Indeed, a stronger results holds. Namely, the union is disjoint:

Lemma 3.13. $\mathsf{names}(\rho\sigma) = \mathsf{names}(\rho) \uplus \mathsf{names}(\sigma)$

Proof. Let $\rho : t \twoheadrightarrow_{\#} s$. By induction on ρ we argue that the set of labels decorating the lambdas in s is disjoint from $\mathsf{names}(\rho)$. The base case is immediate, so let $\rho = T\tau$. The step T is of the form:

$$T \quad : \quad t = \mathtt{C}\langle(\lambda^{\ell} x.u)\vec{r}\rangle \to_{\#} \mathtt{C}\langle u\{x := \vec{r}\}\rangle$$

hence the target of T has no lambdas decorated with the label ℓ. Moreover, the derivation τ is of the form:

$$\tau \quad : \quad \mathtt{C}\langle u\{x := \vec{r}\}\rangle \twoheadrightarrow_{\#} s$$

and by *i.h.* the set of labels decorating the lambdas in s is disjoint from $\mathsf{names}(\tau)$. As a consequence, the set of labels decorating the lambdas in s is disjoint from both $\mathsf{names}(\tau)$ and $\{\ell\}$. This completes the proof. □

Corollary 3.14 (The length of a derivation equals the number of distinct names along it)**.** *If ρ is a derivation in the distributive lambda-calculus and $|\rho|$ denotes the length of ρ, then* $|\rho| = \#(\mathsf{names}(\rho))$.

Proof. Let $\rho = R_1 \dots R_n$. Then by Lemma 3.13, $\mathsf{names}(\rho) = \biguplus_{i=1}^{n} \{\mathsf{name}(R_i)\}$ and $\#(\mathsf{names}(\rho)) = n$, as required. □

The names of a derivation are coherent and work well with all the usual residual theory definitions: projections, prefix order, and permutation equivalence.

Lemma 3.15 (Names after a projection)**.** *If ρ and σ are coinitial derivations, then* $\mathsf{names}(\rho/\sigma) = \mathsf{names}(\rho) \setminus \mathsf{names}(\sigma)$

Proof. First we claim that $\mathsf{names}(\rho/R) = \mathsf{names}(\rho) \setminus \{\mathsf{name}(R)\}$. We proceed by induction on ρ. The base case is immediate, so let $\rho = S\sigma$ and consider two sub-cases, depending on whether $R = S$. If $R = S$, then:

$$\begin{array}{rcl} \mathsf{names}(\rho/R) & = & \mathsf{names}(R\sigma/R) \\ & = & \mathsf{names}(\sigma) \\ & = & \mathsf{names}(R\sigma) \setminus \{\mathsf{name}(R)\} \end{array}$$

On the other hand if $R \neq S$, then by Lemma 3.5 we have that $R/S = R'$ where $\mathsf{name}(R) = \mathsf{name}(R')$ and, similarly, $S/R = S'$, where $\mathsf{name}(S) = \mathsf{name}(S')$. Hence:

$$\begin{array}{rcll} \mathsf{names}(\rho/R) & = & \mathsf{names}(S\sigma/R) & \\ & = & \mathsf{names}((S/R)(\sigma/(R/S))) & \\ & = & \mathsf{names}(S/R) \cup \mathsf{names}(\sigma/(R/S)) & \\ & = & \mathsf{names}(S') \cup \mathsf{names}(\sigma/R') & \\ & = & \mathsf{names}(S') \cup (\mathsf{names}(\sigma) \setminus \{\mathsf{name}(R')\} & \text{by } \textit{i.h.} \\ & = & \mathsf{names}(S) \cup (\mathsf{names}(\sigma) \setminus \{\mathsf{name}(R)\}) & \\ & = & (\mathsf{names}(S) \cup \mathsf{names}(\sigma)) \setminus \{\mathsf{name}(R)\} & \text{since } \mathsf{name}(R) \neq \mathsf{name}(S) \\ & = & \mathsf{names}(S\sigma) \setminus \{\mathsf{name}(R)\} & \\ & = & \mathsf{names}(\rho) \setminus \{\mathsf{name}(R)\} & \end{array}$$

which completes the claim. To see that $\mathsf{names}(\rho/\sigma) = \mathsf{names}(\rho) \setminus \mathsf{names}(\sigma)$ for an arbitrary derivation σ, proceed by induction on σ. If σ is empty it is trivial, so consider the case in which $\sigma = R\tau$. Then:

$$\begin{array}{rcll} \mathsf{names}(\rho/R\tau) & = & \mathsf{names}((\rho/R)/\tau) & \\ & = & \mathsf{names}(\rho/R) \setminus \mathsf{names}(\tau) & \text{by } \textit{i.h.} \\ & = & (\mathsf{names}(\rho) \setminus \{\mathsf{name}(R)\}) \setminus \mathsf{names}(\tau) & \text{by the previous claim} \\ & = & \mathsf{names}(\rho) \setminus (\{\mathsf{name}(R)\} \cup \mathsf{names}(\tau)) & \\ & = & \mathsf{names}(\rho) \setminus \mathsf{names}(R\tau) & \end{array}$$

as required. □

Proposition 3.16 (Prefixes as subsets)**.** *Let ρ, σ be coinitial derivations in the distributive lambda-calculus. Then $\rho \sqsubseteq \sigma$ if and only if $\mathsf{names}(\rho) \subseteq \mathsf{names}(\sigma)$.*

Proof. By induction on ρ. The base case is immediate since $\epsilon \sqsubseteq \sigma$ and $\varnothing \subseteq \mathsf{names}(\sigma)$ both hold. So let $\rho = T\tau$. First note that the following equivalence holds:

$$T\tau \sqsubseteq \sigma \quad \iff \quad T \sqsubseteq \sigma \ \wedge \ \tau \sqsubseteq \sigma/T \tag{1}$$

Indeed:

- ($\Rightarrow$) Suppose that $T\tau \sqsubseteq \sigma$. Then, on one hand, $T \sqsubseteq T\tau \sqsubseteq \sigma$. On the other hand, projection is monotonic, so $\tau = T\tau/T \sqsubseteq \sigma/T$.
- ($\Leftarrow$) Since $\tau \sqsubseteq \sigma/T$ we have that $T\tau \sqsubseteq T(\sigma/T) \equiv \sigma(T/\sigma) = \sigma$ since $T/\sigma = \epsilon$.

So we have that:

$$\begin{array}{rcll} T\tau \sqsubseteq \sigma & \iff & T \sqsubseteq \sigma \ \wedge \ \tau \sqsubseteq \sigma/T & \text{by (1)} \\ & \iff & \mathsf{name}(T) \in \mathsf{names}(\sigma) \ \wedge \ \tau \sqsubseteq \sigma/T & \text{by Lemma 3.11} \\ & \iff & \mathsf{name}(T) \in \mathsf{names}(\sigma) \ \wedge \ \mathsf{names}(\tau) \subseteq \mathsf{names}(\sigma/T) & \text{by } \textit{i.h.} \\ & \iff & \mathsf{name}(T) \in \mathsf{names}(\sigma) \ \wedge \ \mathsf{names}(\tau) \subseteq \mathsf{names}(\sigma) \setminus \{\mathsf{name}(T)\} & \text{by Lemma 3.15} \\ & \iff & \mathsf{names}(T\tau) \subseteq \mathsf{names}(\sigma) & \end{array}$$

To justify the very last equivalence, the ($\Rightarrow$) direction is immediate. For the ($\Leftarrow$) direction, the difficulty is ensuring that $\mathsf{names}(\tau) \subseteq \mathsf{names}(\sigma) \setminus \{\mathsf{name}(T)\}$ from the fact that $\mathsf{names}(T\tau) \subseteq \mathsf{names}(\sigma)$. To see this it suffices to observe that by Lemma 3.13, $\mathsf{names}(T\tau)$ is the *disjoint* union $\mathsf{names}(T) \uplus \mathsf{names}(\tau)$, which means that $\mathsf{name}(T) \notin \mathsf{names}(\tau)$.

□

Corollary 3.17 (Permutation equivalence in terms of names)**.** *Let ρ, σ be coinitial derivations in the distributive lambda-calculus. Then $\rho \equiv \sigma$ if and only if $\mathsf{names}(\rho) = \mathsf{names}(\sigma)$.*

Proof. Immediate since

$$\begin{array}{rcll} \rho \equiv \sigma & \iff & \rho \sqsubseteq \sigma \ \wedge \ \sigma \sqsubseteq \rho & \\ & \iff & \mathsf{names}(\rho) \subseteq \mathsf{names}(\sigma) \ \wedge \ \mathsf{names}(\sigma) \subseteq \mathsf{names}(\rho) & \text{by Proposition 3.16} \\ & \iff & \mathsf{names}(\rho) = \mathsf{names}(\sigma) & \end{array}$$

□

3.3 Stability (and creation)

In this section we will see two important lemmas that guarantee certain properties of residues after steps or derivations.

The result that we will call full stability can be seen as a functorial extension to the confluence result: confluence speaks about happens to terms after two coinitial derivations, while full stability speaks about what happens to steps after two coinitial derivations.

Creation

The first of this lemmas is creation. We say that a step R is **created** by another step S if there is no step R' coinitial with S such that $R \in R'/S$. What this means is that we weren't able to do what R does before executing S. For example, in Example 3.8 we claimed that the step T is created by R, which indeed is.

Having a lemma that characterizes creation is relevant because in proofs, it is common to have edge cases where a new redex is created. A creation lemma can help handle those edge cases more easily.

The following lemma says that if S creates R, then S follows one of three general forms.

Lemma 3.18 (Creation). *There are three creation cases in the distributive lambda-calculus:*

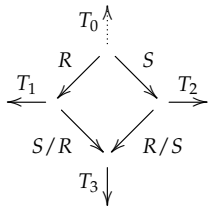
1. **Creation case I.** $\mathtt{C}\langle(\lambda^{\ell} x.x^{\tau})\,[\lambda^{\ell'} y.t]\,\vec{s}\rangle \to_{\#} \mathtt{C}\langle(\lambda^{\ell'} y.t)\,\vec{s}\rangle \to_{\#} \mathtt{C}\langle t\{\!\{y := \vec{s}\}\!\}\rangle$.
2. **Creation case II.** $\mathtt{C}\langle(\lambda^{\ell} x.\lambda^{\ell'} y.t)\,\vec{s}\,\vec{u}\rangle \to_{\#} \mathtt{C}\langle(\lambda^{\ell'} y.t')\,\vec{u}\rangle \to_{\#} \mathtt{C}\langle t'\{\!\{y := \vec{u}\}\!\}\rangle$, *where:*
$$t' \;=\; t\{\!\{x := \vec{s}\}\!\}$$
3. **Creation case III.** $\mathtt{C}_1\langle(\lambda^{\ell} x.\mathtt{C}_2\langle x^{\tau}\,\vec{t}\rangle)\vec{s}\rangle \to_{\#} \mathtt{C}_1\langle\mathtt{C}'_2\langle(\lambda^{\ell'} y.u)\,\vec{t'}\rangle\rangle \to_{\#} \mathtt{C}_1\langle\mathtt{C}'_2\langle u\{\!\{y := \vec{t'}\}\!\}\rangle\rangle$, *where:*
$$\begin{aligned} \mathtt{C}_2\{\!\{x := \vec{s}\}\!\} &= \mathtt{C}'_2 \\ x^{\tau}\{\!\{x := \vec{s}\}\!\} &= \lambda^{\ell'} y.u \\ \vec{t}\{\!\{x := \vec{s}\}\!\} &= \vec{t'} \\ \vec{s} &= [\vec{s_1}, \lambda^{\ell'} y.u, \vec{s_2}] \end{aligned}$$

Proof. Let $R : \mathtt{C}\langle(\lambda^{\ell} x.t)\vec{s}\rangle \to_{\#} \mathtt{C}\langle t\{\!\{x := \vec{s}\}\!\}\rangle$ be a step, and let $S : \mathtt{C}\langle t\{\!\{x := \vec{s}\}\!\}\rangle \to_{\#} p$ another step such that R creates S. The redex contracted by the step S is below a context $\mathtt{C}_1$, so let $\mathtt{C}\langle t\{\!\{x := \vec{s}\}\!\}\rangle = \mathtt{C}_1\langle(\lambda^{\ell'} y.u)\vec{r}\rangle$, where $(\lambda^{\ell'} y.u)\vec{r}$ is the redex contracted by S. We need consider three cases, depending on the relative positions of the holes of $\mathtt{C}$ and $\mathtt{C}_1$, namely they may be disjoint, $\mathtt{C}$ may be a prefix of $\mathtt{C}_1$, or $\mathtt{C}_1$ may be a prefix of $\mathtt{C}$. ♠ □

Stability

We also would like to prove a property that we call *full stability*, which is a strong version of stability in the sense of Lévy [Lev15] (which can in turn be traced back to Berry's notion of stability).

Recall that Lévy's stability states that for any $R \neq S$, if T_1 and T_2 have a common descendant T_3, as in the figure below, then they have a common ancestor T_0.

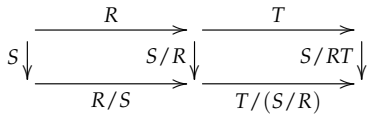


In the figure above, if T_1 and T_2 have a common descendant T_3 means that $T_3 \in T_1/(S/R)$ and $T_3 \in T_2/(R/S)$. Informally, this means that T_1 and T_2 do the same work, which means that that work is enabled by either performing R or S. The fact that a system is stable means that we could actually perform the work that T_1 and T_2 do, without executing R nor S (by doing T_0).

Hence, what stability means is that steps are created essentially in a unique way.

Lemma 3.19 (Stability)**.** *If R, S are different coinitial steps such that R creates a step T, then R/S creates the step $T/(S/R)$.*

Diagrammatically:

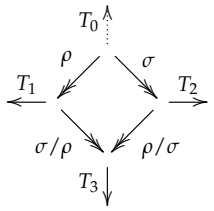


Note: *It is easy to see that this is equivalent to stability in the sense of Lévy.*

Proof. Let $R : \mathtt{C}\langle(\lambda^{\ell} x.t)\vec{s}\rangle \to_{\#} \mathtt{C}\langle t\{x := \vec{s}\}\rangle$, let $S \neq R$ be a step coinitial to R, and suppose that R creates a step T. By induction on the context $\mathtt{C}$ we argue that R/S creates $T/(S/R)$. □

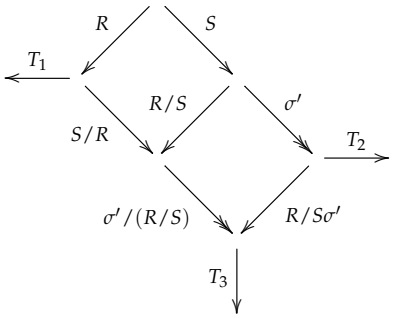
Next we prove *full stability*, which is an extension to Lévy's notion of stability, from steps to arbitrary reductions. Informally, it says that if we are able to execute the same step after two completely different derivations, then the step existed before both reductions—*i.e.*, a step is only created by a single reduction (modulo permutation equivalence).

Lemma 3.20 (Full stability)**.** *Let ρ and σ be coinitial derivations such that* $\mathsf{names}(\rho) \cap \mathsf{names}(\sigma) = \varnothing$*. Let T_1, T_2, and T_3 be steps such that $T_3 = T_1/(\sigma/\rho) = T_2/(\rho/\sigma)$. Then there exists a step T_0 such that $T_1 = T_0/\rho$ and $T_2 = T_0/\sigma$. Diagrammatically:*



Proof. We first prove the proposition in the particular case in which ρ is a single step, *i.e.* $\rho = R$ By induction on σ:

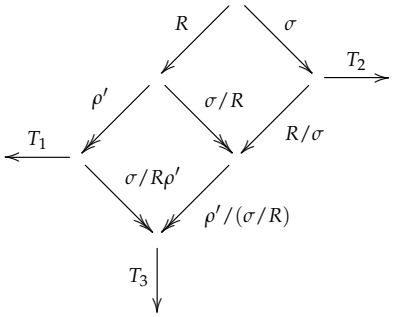
1. **Empty,** $\sigma = \epsilon$**.** Then $T_1 = T_3 = T_2/R$, so it suffices to take $T_0 := T_2$.
2. **Non-empty,** $\sigma = S\sigma'$**.** Recall that permutation diagrams in the distributive lambda-calculus are *square*, (*i.e.* steps always have exactly one residual, except for the trivial case $R/R = \varnothing$, as was proved in Lemma 3.5). Since $\mathsf{name}(R) \notin \mathsf{names}(S\sigma')$, we know that R is not any of the steps along $S\sigma'$, so in particular R/S and $R/S\sigma'$ are singletons, which means that the situation is the following:



Observe that $T_3 = T_1/((S/R)(\sigma'/(R/S)))$, so by taking $T_1' := T_1/(S/R)$ we have that $T_3 = T_1'/(\sigma'/(R/S))$. By *i.h.* on σ' we have that there exists a step T_2' such that $T_1' = T_2'/(R/S)$ and $T_2 = T_2'/\sigma'$.
To conclude, note that $R \neq S$ and $T_1' = T_1/(S/R) = T_0/(R/S)$ so by the Stability lemma (Lemma 3.19) there must exist a step T_0 such that $T_1 = T_0/R$ and $T_2' = T_0/S$. Moreover $T_2 = T_2'/\sigma' = T_0/S\sigma'$, as required.

Having established the previous claim, let us now prove the main statement of the proposition by induction on ρ.

1. **Empty,** $\rho = \epsilon$**.** Then $T_2 = T_3 = T_1/\sigma$ so by taking $T_0 := T_1$ we conclude.
2. **Non-empty,** $\rho = R\rho'$**.** First observe that, since $\mathsf{names}(R\rho') \cap \mathsf{names}(\sigma) = \varnothing$, we have that $\mathsf{names}(R) \cap \mathsf{names}(\sigma) = \varnothing$ and $\mathsf{names}(\rho') \cap \mathsf{names}(\sigma) = \varnothing$. The situation is the following:



Observe that $T_3 = T_2/((R/\sigma)(\rho'/(\sigma/R)))$, so by taking $T_2' := T_2/(R/\sigma)$ we have that $T_3 = T_2'/(\rho'/(\sigma/R))$. By Lemma 3.15 we have that $\mathsf{names}(\sigma/R) = \mathsf{names}(\sigma) \setminus \{\mathsf{name}(R)\}$ and $\mathsf{name}(R) \notin \mathsf{names}(\sigma)$, so $\mathsf{names}(\sigma/R) = \mathsf{names}(\sigma)$. In particular, $\mathsf{names}(\rho') \cap \mathsf{names}(\sigma/R) = \varnothing$ so we may apply the *i.h.* on ρ' to conclude that there exists a step T_1' such that $T_1 = T_1'/\rho'$ and $T_2' = T_1'/(\sigma/R)$.

To conclude, observe that $T_2' = T_1'/(\sigma/R) = T_2/(R/\sigma)$, where $\mathsf{name}(R) \notin \mathsf{names}(\sigma)$, so by the previous claim we have that there is a step T_0 such that $T_0/R = T_1'$ and $T_0/\sigma = T_2$. Moreover, $T_0/R\rho' = T_1'/R = T_1$ so we are done.

□

3.4 Lattices and derivation spaces

In this section we are going to consider the derivation spaces of terms in the $\lambda^{\#}$-calculus as lattices and prove properties about those lattices.

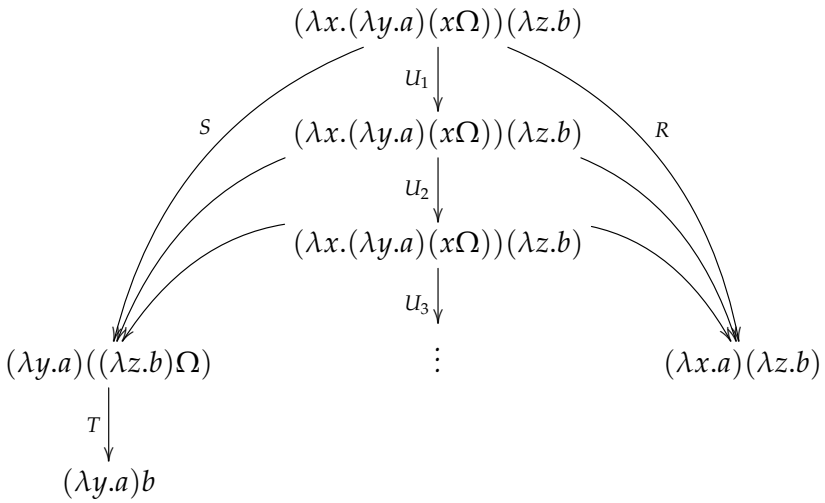
In a general orthogonal rewrite system, the **space of derivations** of a term t is the set $\{\rho : \mathsf{src}(\rho) = t\}$ (usually modulo permutation equivalence), together with the order relation $\sqsubseteq$. Recall that $\rho \sqsubseteq \sigma$ if and only if $\rho/\sigma = \epsilon$.

In the preliminaries we defined a **lattice** to be partially ordered set with two operations: join and meet. Given two elements of the lattice, their join is their least upper bound, and their join is their greatest lower bound. Notice that those elements have to be unique.

As a consequence of ~~and~~ axiomatic results [Mel96], derivation spaces of terms of orthogonal abstract rewrite systems are *upper semilattices*. This means that they have joins, but not necessarily meets.

Joins for these systems are given essentially by their confluence property: the join (or least upper bound) of two reductions ρ, σ is $\rho(\sigma/\rho)$ (or equivalently $\sigma(\rho/\sigma)$). This can be rephrased in category theory by saying that the derivation space of terms of orthogonal abstract rewrite systems have push-outs.

Example 3.21. *Laneve's counterexample from the introduction showed that the meet of derivations is not well defined for the pure lambda calculus (which is a orthogonal abstract rewrite system [Mel96]). Let $\Omega = (\lambda x.xx)\lambda x.xx$, and consider the reduction space of $(\lambda x.(\lambda y.a)(x\Omega))(\lambda z.b)$ where the steps U_i contract Ω:*



As we pointed out in the introduction, ST and R do not have a greatest lower bound: ST and R are not comparable, and neither are S and R, and for all $n \in \mathbb{N}$ we have that $U_1 \dots U_n \sqsubseteq R$ and $U_1 \dots U_n \sqsubseteq ST$ so the meet $R \sqcap ST$ does not exist.

Let us write $\mathbb{D}^{\#}(t)$ for the set of derivations of t in the $\lambda^{\#}$-calculus, modulo permutation equivalence. Because of Orthogonality (Proposition 3.7) the set $\mathbb{D}^{\#}(t)$ armed with $\sqsubseteq$ and $\sqcup$ is an upper semilattice. Following, we prove that moreover the space $\mathbb{D}^{\#}(t)$ is a *distributive lattice*.

Definition 3.22. A **distributive lattice** is a lattice $(A, \leq, \vee, \wedge)$ such that for all $a, b, c \in A$, $a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c)$ and $a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c)$. That is, each operation is distributive over the other.

Example 3.23. *For example, if X is a set, then $(\mathcal{P}(X), \subseteq, \cup, \cap)$, its powerset, is a distributive lattice.*

If we want to prove that $\mathbb{D}^{\#}(t)$ is a distributive lattice, we need to prove that it is a lattice and that it satisfies the distributivity property. We know that $\mathbb{D}^{\#}(t)$ is a join semilattice (or upper semilattice), so to prove that it is a lattice we need to define a meet that works.

Meets

The informal idea behind the definition below is that the meet of two derivations ρ and σ will be the work they both have performed.

Proposition 3.24 (Meet of derivations)**.** *Let ρ, σ be coinitial derivations in the distributive lambda-calculus. Then there exists an infimum for ρ, σ with respect to the prefix order $\sqsubseteq$. We write $\rho \sqcap \sigma$ for the infimum of $\{\rho, \sigma\}$ obtained by this construction.*

Proof. If ρ and σ are derivations, we say that a step R is a **common** (to ρ and σ) whenever $R \in \rho$ and $R \in \sigma$. Define $\rho \sqcap \sigma$ as follows, by induction on the length of ρ:

$$\rho \sqcap \sigma \stackrel{\text{def}}{=} \begin{cases} \epsilon & \text{if there are no common steps to } \rho \text{ and } \sigma \\ R((\rho/R) \sqcap (\sigma/R)) & \text{if the step } R \text{ is common to } \rho \text{ and } \sigma \end{cases}$$

In the second case of the definition, there might be more than one R common to ρ and σ. We suppose that one of them is chosen deterministically but make no further assumptions. To see that this recursive construction is well-defined, note that the length of ρ/R is lesser than the length of ρ by the fact that projections are decreasing (Lemma A.10). To conclude the construction, we show that $\rho \sqcap \sigma$ is an infimum, *i.e.* a greatest lower bound:

1. **Lower bound.** Let us show that $\rho \sqcap \sigma \sqsubseteq \rho$ by induction on the length of ρ. There are two sub-cases, depending on whether there is a step common to ρ and σ.
   If there is no common step, then $\rho \sqcap \sigma = \epsilon$ trivially verifies $\rho \sqcap \sigma \sqsubseteq \rho$.
   On the other hand, if there is a common step, we have by definition that $\rho \sqcap \sigma = R((\rho/R) \sqcap (\sigma/R))$ where R is common to ρ and σ. Recall that projections are decreasing (Lemma A.10)

so $|\rho| > |\rho/R|$. This allows us to apply the *i.h.* and conclude:

$$\begin{array}{rcll} \rho \sqcap \sigma & = & R((\rho/R) \sqcap (\sigma/R)) & \text{by definition} \\ & \sqsubseteq & R(\rho/R) & \text{since by } \textit{i.h.}\ (\rho/R) \sqcap (\sigma/R) \sqsubseteq \rho/R \\ & \equiv & \rho(R/\rho) & \\ & = & \rho & \text{since } R \sqsubseteq \rho \text{ by Lemma 3.11.} \end{array}$$

Showing that $\rho \sqcap \sigma \sqsubseteq \sigma$ is symmetric, by induction on the length of σ.

2. **Greatest lower bound.** Let τ be a lower bound for $\{\rho, \sigma\}$, *i.e.* $\tau \sqsubseteq \rho$ and $\tau \sqsubseteq \sigma$, and let us show that $\tau \sqsubseteq \rho \sqcap \sigma$. We proceed by induction on the length of ρ. There are two sub-cases, depending on whether there is a step common to ρ and σ.
If there is no common step, we claim that τ must be empty. Otherwise we would have that $\tau = T\tau' \sqsubseteq \rho$ so in particular $T \sqsubseteq \rho$ and $T \in \rho$ by Lemma 3.11. Similarly, $T \in \sigma$ so T is a step common to ρ and σ, which is a contradiction. We obtain that τ is empty, so trivially $\tau = \epsilon \sqsubseteq \rho \sqcap \sigma$.
On the other hand, if there is a common step, we have by definition that $\rho \sqcap \sigma = R((\rho/R) \sqcap (\sigma/R))$ where R is common to ρ and σ. Moreover, since $\tau \sqsubseteq \rho$ and $\tau \sqsubseteq \sigma$, by projecting along R we know that $\tau/R \sqsubseteq \rho/R$ and $\tau/R \sqsubseteq \sigma/R$. So:

$$\begin{array}{rcll} \tau & \sqsubseteq & \tau(R/\tau) & \\ & \equiv & R(\tau/R) & \\ & \sqsubseteq & R((\rho/R) \sqcap (\sigma/R)) & \text{since by } \textit{i.h.}\ \tau/R \sqsubseteq (\rho/R) \sqcap (\sigma/R) \\ & = & \rho \sqcap \sigma & \text{by definition} \end{array}$$

□

Remark 3.25. The infimum of $\{\rho, \sigma\}$ is unique modulo permutation equivalence, *i.e.* if τ is an infimum for $\{\rho, \sigma\}$ then $\tau \equiv \rho \sqcap \sigma$.

Names of join and meet

Names of derivations will be a helpful tool when computing meets and joins of derivations, as the next proposition allows us to easily get them by taking the intersection or union of sets.

Proposition 3.26 (Names of join and meet)**.** *The following hold:*

1. $\mathsf{names}(\rho \sqcup \sigma) = \mathsf{names}(\rho) \cup \mathsf{names}(\sigma)$
2. $\mathsf{names}(\rho \sqcap \sigma) = \mathsf{names}(\rho) \cap \mathsf{names}(\sigma)$

Proof. Item 1 is easy resorting to the definition of $\sqcup$ and Lemma 3.15:

$$\begin{array}{rcll} \mathsf{names}(\rho \sqcup \sigma) & = & \mathsf{names}(\rho(\sigma/\rho)) & \text{by definition of } \sqcup \\ & = & \mathsf{names}(\rho) \cup \mathsf{names}(\sigma/\rho) & \\ & = & \mathsf{names}(\rho) \cup (\mathsf{names}(\sigma) \setminus \mathsf{names}(\rho)) & \text{by Lemma 3.15} \\ & = & \mathsf{names}(\rho) \cup \mathsf{names}(\sigma) & \end{array}$$

For item 2., the inclusion $\mathsf{names}(\rho \sqcap \sigma) \subseteq \mathsf{names}(\rho) \cap \mathsf{names}(\sigma)$ is an immediate consequence of Lemma A.12. For other inclusion, namely to show $\mathsf{names}(\rho) \cap \mathsf{names}(\sigma) \subseteq \mathsf{names}(\rho \sqcap \sigma)$, we first prove the following claim:

Claim. $\mathsf{names}(\rho/(\rho \sqcap \sigma)) \cap \mathsf{names}(\sigma/(\rho \sqcap \sigma) = \varnothing$. *Proof of the claim.* By Lemma A.11 it suffices to show that $(\rho/(\rho \sqcap \sigma)) \sqcap (\sigma/(\rho \sqcap \sigma)) = \epsilon$. By contradiction, suppose that there is a step

T common to the derivations $\rho/(\rho \sqcap \sigma)$ and $\sigma/(\rho \sqcap \sigma)$. Then the derivation $(\rho \sqcap \sigma)T$ is a lower bound for $\{\rho, \sigma\}$, *i.e.* $(\rho \sqcap \sigma)T \sqsubseteq \rho$ and $(\rho \sqcap \sigma)T \sqsubseteq \sigma$. Since $\rho \sqcap \sigma$ is the greatest lower bound for $\{\rho, \sigma\}$, we have that $(\rho \sqcap \sigma)T \sqsubseteq \rho \sqcap \sigma$. But this implies that $T \sqsubseteq \epsilon$, which is a contradiction. This concludes the proof of the claim.

Note that $\rho \sqcap \sigma \sqsubseteq \rho$, so we have that $\rho \equiv (\rho \sqcap \sigma)(\rho/(\rho \sqcap \sigma))$, and this in turn implies that $\mathsf{names}(\rho) = \mathsf{names}((\rho \sqcap \sigma)(\rho/(\rho \sqcap \sigma)))$ by Corollary 3.17. Symmetrically, $\mathsf{names}(\sigma) = \mathsf{names}((\rho \sqcap \sigma)(\sigma/(\rho \sqcap \sigma)))$. Then:

$$\begin{aligned}
& \mathsf{names}(\rho) \cap \mathsf{names}(\sigma) \\
= \;& \mathsf{names}((\rho \sqcap \sigma)(\rho/(\rho \sqcap \sigma))) \cap \mathsf{names}((\rho \sqcap \sigma)(\sigma/(\rho \sqcap \sigma))) \\
= \;& (\mathsf{names}(\rho \sqcap \sigma) \cup \mathsf{names}(\rho/(\rho \sqcap \sigma))) \cap (\mathsf{names}(\rho \sqcap \sigma) \cup \mathsf{names}(\sigma/(\rho \sqcap \sigma))) \\
& \text{by Remark 3.12} \\
= \;& \mathsf{names}(\rho \sqcap \sigma) \cup (\mathsf{names}(\rho/(\rho \sqcap \sigma)) \cap \mathsf{names}(\sigma/(\rho \sqcap \sigma))) \\
& \text{since } (A \cup B) \cap (A \cup C) = A \cup (B \cap C) \text{ for arbitrary sets } A, B, C \\
= \;& \mathsf{names}(\rho \sqcap \sigma) \\
& \text{since } (\mathsf{names}(\rho/(\rho \sqcap \sigma)) \cap \mathsf{names}(\sigma/(\rho \sqcap \sigma))) = \varnothing \text{ by the previous claim}
\end{aligned}$$

This concludes the proof. □

Distributive lattice

Theorem 3.27 (Derivations modulo $\equiv$ form a distributive lattice)**.** *Let $t \in \mathcal{T}^{\#}$ be a correct term. Then derivations modulo $\equiv$ form a lattice $\mathbb{D}^{\#}(t)$. More precisely, let X be the set of derivations in the distributive lambda-calculus going out from t, modulo permutation equivalence:*

$$X \stackrel{\text{def}}{=} \frac{\{\rho \mid \mathsf{src}(\rho) = t\}}{\equiv}$$

Let moreover $[\rho]$ denote the equivalence class of a derivation ρ modulo $\equiv$. Then $\mathbb{D}^{\#}(t) \stackrel{\text{def}}{=} (X, \leq, \wedge, \vee)$ is a distributive lattice, where:

$$\begin{aligned}
[\rho] \leq [\sigma] &\iff^{\text{def}} \rho \sqsubseteq \sigma \\
[\rho] \wedge [\sigma] &\iff^{\text{def}} \rho \sqcap \sigma \\
[\rho] \vee [\sigma] &\iff^{\text{def}} \rho \sqcup \sigma
\end{aligned}$$

Proof. It is straightforward to check that $\leq$ is a partial order, and that $[\rho] \wedge [\sigma]$ (resp. $[\rho] \vee [\sigma]$) is the infimum (resp. supremum) of $\{[\rho], [\sigma]\}$.

To see that it is distributive, let us first prove the first distributive law: $([\rho] \wedge [\sigma]) \vee [\tau] = ([\rho] \vee [\tau]) \wedge ([\sigma] \vee [\tau])$. Let ρ, σ, τ be arbitrary coinitial derivations. The following equality holds trivially, since $(A \cap B) \cup C = (A \cup C) \cap (A \cup B)$ is valid for arbitrary sets A, B, C:

$$(\mathsf{names}(\rho) \cap \mathsf{names}(\sigma)) \cup \mathsf{names}(\tau) = (\mathsf{names}(\rho) \cup \mathsf{names}(\tau)) \cap (\mathsf{names}(\sigma) \cup \mathsf{names}(\tau))$$

By Proposition 3.26 this entails:

$$\mathsf{names}((\rho \sqcap \sigma) \sqcup \tau) = \mathsf{names}((\rho \sqcup \tau) \sqcap (\sigma \sqcup \tau))$$

By Corollary 3.17 this in turn implies that:

$$(\rho \sqcap \sigma) \sqcup \tau \equiv (\rho \sqcup \tau) \sqcap (\sigma \sqcup \tau)$$

So by definition of $\wedge, \vee$ we obtain:

$$([\rho] \wedge [\sigma]) \vee [\tau] = ([\rho] \vee [\tau]) \wedge ([\sigma] \vee [\tau])$$

The other distributive law, namely $([\rho] \vee [\sigma]) \wedge [\tau] = ([\rho] \wedge [\tau]) \vee ([\sigma] \wedge [\tau])$ is proved analogously. □

Remark 3.28. The function names that takes a derivation and returns a set of labels is well-defined for permutation-equivalence classes, as a consequence of Corollary 3.17:

$$\mathsf{names}([\rho]) \stackrel{\text{def}}{=} \mathsf{names}(\rho)$$

Theorem 3.29 (The lattice of derivations is representable as a ring of sets)**.** *If $t \in \mathcal{T}^{\#}$ is a correct term, then* $\mathsf{names} : \mathbb{D}^{\#}(t) \to \mathcal{P}(\mathscr{L})$ *is a monomorphism of lattices, where $\mathbb{D}^{\#}(t)$ is the lattice of derivations of t and $\mathcal{P}(\mathscr{L})$ is the lattice whose elements are sets of labels ordered by inclusion, with set intersection and set union as the meet and join operators.*

Proof. We are to show that names is monotonic, that it preserves meets and joins, and finally that it is a monomorphism:

- **Monotonic.** If $[\rho] \leq [\sigma]$ then $\mathsf{names}(\rho) \subseteq \mathsf{names}(\sigma)$. This has been proved in Proposition 3.16.
- **Preserves meets.** $\mathsf{names}([\rho] \wedge [\sigma]) = \mathsf{names}(\rho) \cap \mathsf{names}(\sigma)$ by Proposition 3.26.
- **Preserves joins.** $\mathsf{names}([\rho] \vee [\sigma]) = \mathsf{names}(\rho) \cup \mathsf{names}(\sigma)$ by Proposition 3.26.
- **Monomorphism.** It suffices to show that names is injective. Indeed, suppose that $\mathsf{names}([\rho]) = \mathsf{names}([\sigma])$. By Corollary 3.17 we have that $\rho \equiv \sigma$, so $[\rho] = [\sigma]$.

□

Example 3.30. *If we consider the derivation space from Example 3.8, we see that we have 8 derivations modulo permutation equivalence, representable by the powerset of the set of labels $\{1, 2, 3\}$. Some examples:*

- *$\varnothing$ represents the empty derivation, ϵ.*
- *$\{1\}$ represents R.*
- *$\{1, 2, 3\}$ represents $[RS'T'']$, which is equal to $[RTS'']$ and $[SR'T']$.*
- *As $\{1\}$ represents R and $\{3\}$ represents S, and $\{1\} \cup 3 = \{1, 3\}$, we know that $R \sqcup S = [RS']$.*

Chapter 4

Simulation of the λ-calculus

In this chapter we will focus on establishing a correspondance between the λ-calculus and the $\lambda^{\#}$-calculus. Our goal is to be able to prove that we can map derivations in one calculus to derivations in the other.

First, we will learn how to relate terms from one calculus to terms in the other, via what we call *refinements*. Then, we will look at how the derivation spaces of two related terms are connected. After that we will learn that the fact that a term of the lambda calculus is related to another in the distributive lambda calculus says that it has a head normal form. Finally, we will link and combine the residual theories of both calculi.

Notation 4.1. If t is a term of the distributive lambda calculus, we already defined $\mathbb{D}^{\#}(t)$ to be its derivation space, which we proved is a distributive lattice. $\mathcal{T}^{\#}$ is the set of all (correct) terms.

If s is a term of the pure lambda calculus, we write $\mathbb{D}^{\lambda}(s)$ to mean the derivation space of s, which is an join semilattice. $\mathcal{T}^{\lambda}$ is the set of all terms of the pure lambda calculus.

4.1 Refinements

Definition 4.2 (Refinement)**.** A lambda-term $t \in \mathcal{T}^{\lambda}$ is **refined** by a distributive term $t' \in \mathcal{T}^{\#}$, written $t \rtimes t'$ according to the following inductive definition:

$$\frac{}{x \rtimes x^{\tau}}\ \texttt{r-var} \qquad \frac{t \rtimes t'}{\lambda x.t \rtimes \lambda^{\ell} x.t'}\ \texttt{r-lam} \qquad \frac{t \rtimes t' \quad s \rtimes s_i \text{ for all } i = 1..n}{t\,s \rtimes t'[s_1, \ldots, s_n]}\ \texttt{r-app}$$

We write $t' \ltimes t$ if $t \rtimes t'$. Refinement is also generalized to contexts, by declaring that $\Box \ltimes \Box$.

Note that a term can have more than one refinement. For example, the following terms refine $(\lambda x.xx)y$:

$$(\lambda^1 x.x^{[\,]\xrightarrow{2}\alpha^3}[\,])[y^{[\,]\xrightarrow{2}\alpha^3}] \qquad (\lambda^1 x.x^{[\alpha^2,\beta^3]\xrightarrow{4}\gamma^5}[x^{\alpha^2}, x^{\beta^3}])[y^{[\alpha^2,\beta^3]\xrightarrow{4}\gamma^5}, y^{\alpha^2}, y^{\beta^3}]$$
$$(\lambda^1 x.x^{[\alpha^2]\xrightarrow{3}\beta^4}[x^{\alpha^2}])[y^{[\alpha^2]\xrightarrow{3}\beta^4}, y^{\alpha^2}] \qquad (\lambda^1 x.x^{[\alpha^2]\xrightarrow{3}\beta^4}[x^{\alpha^2}])[y^{\alpha^2}, y^{[\alpha^2]\xrightarrow{3}\beta^4}]$$

Also, a term may have no refinements—for example, $\Omega = (\lambda x.xx)(\lambda x.xx)$. We leave this as an exercise to the reader.

The fact that a term doesn't have a refinement doesn't imply that sub-terms or terms that contain it as a sub-term don't have refinements. For example $\lambda x.xx$ is refinable by $\lambda^{\ell} x.x^{[]\to\alpha}[]$, and $(\lambda x.a)\Omega$ is refinable by $(\lambda^{\ell} x.a^{\alpha})[]$.

4.2 Simulation

Having defined the correspondance between terms we have to somehow show how that correspondance is compatible with the notions of reduction in both calculi.

For this, we put forth two simulation results, one that deals with simulating steps of the λ-calculus in the $\lambda^{\#}$-calculus, and another that does the reverse.

Proposition 4.3 (Simulation)**.** *Let $t, s \in \mathcal{T}^{\lambda}$ be lambda-terms and $t' \in \mathcal{T}^{\#}$ be a distributive term such that:*

$$t' \ltimes t \to_{\beta} s$$

then there is a distributive term $s' \in \mathcal{T}^{\#}$ such that:

$$t' \twoheadrightarrow_{\#} s' \ltimes s$$

Diagramatically:

$$\begin{array}{ccc} t & \xrightarrow{\ \beta\ } & s \\ \ltimes & & \ltimes \\ t' & \overset{\#}{\twoheadrightarrow} & s' \end{array}$$

Proof. ♠ By case analysis. The proof is constructive, and the resulting derivation $t' \twoheadrightarrow_{\#} s'$ is a multistep (it is a complete development of a set of coinitial steps). □

Example 4.4. *The following are simulations of the step $x\,((\lambda x.x)y) \to_{\beta} x\,y$ with $\to_{\#}$-steps:*

$$\begin{array}{ccc} x\,((\lambda x.x)y) & \xrightarrow{\ \beta\ } & x\,y \\ \ltimes & & \ltimes \\ x^{[]\overset{1}{\to}\alpha^2}\,[] & = & x^{[]\overset{1}{\to}\alpha^2}\,[] \end{array} \qquad \begin{array}{ccc} x\,((\lambda x.x)y) & \xrightarrow{\qquad\beta\qquad} & x\,y \\ \ltimes & & \ltimes \\ x^{[\alpha^1]\overset{2}{\to}\beta^3}\,[(\lambda^4 x.x^{\alpha^1})[y^{\alpha^1}]] & \xrightarrow{\#} & x^{[\alpha^1]\overset{2}{\to}\beta^3}\,[y^{\alpha^1}] \end{array}$$

$$\begin{array}{ccc} x\,((\lambda x.x)y) & \xrightarrow{\qquad\qquad\beta\qquad\qquad} & x\,y \\ \ltimes & & \ltimes \\ x^{[\alpha^1,\beta^2]\overset{3}{\to}\gamma^4}\,[(\lambda^5 x.x^{\alpha^1})[y^{\alpha^1}], (\lambda^6 x.x^{\beta^2})[y^{\beta^2}]] & \xrightarrow{\#} & x^{[\alpha^1,\beta^2]\overset{3}{\to}\gamma^4}\,[y^{\alpha^1}, y^{\beta^2}] \end{array}$$

Now we want to go the other way: we have $t' \ltimes t$ and $t' \twoheadrightarrow_{\#} s'$. We would like to get a term s such that $t \twoheadrightarrow s \rtimes s'$. But that is not possible in general, to see that consider the following counterexample:

$$\begin{array}{ccc} (\lambda x.xx)((\lambda y.y)a) & \overset{\beta}{\twoheadrightarrow} & s \\ \rtimes & & \rtimes \\ (\lambda^1 x.x^{[\alpha]\to\beta}[x^\alpha])[(\lambda^2 y.y^{[\alpha]\to\beta})a^{[\alpha]\to\beta}, (\lambda 3.yy^\alpha)a^\alpha] & \xrightarrow{\#} & (\lambda^1 x.x^{[\alpha]\to\beta}[x^\alpha])[a^{[\alpha]\to\beta}, (\lambda^3 y.y^\alpha)a^\alpha] \end{array}$$

Note that if s is to be refined by

$$(\lambda^1 x.x^{[\alpha]\to\beta}[x^\alpha])[a^{[\alpha]\to\beta}, (\lambda^3 y.y^\alpha)a^\alpha]$$

then it must be an application, $s = s_1 s_2$, such that s_2 is refined by both $a^{[\alpha]\to\beta}$ and $(\lambda y.y^\alpha)a^\alpha$— but that is impossible, because the first says s_2 should be a variable, and the second says it should be an application. So what we need to reduce the term in the lower right part of the diagram until it refines some term s.

The exact statement for what we want is the proposition that follows.

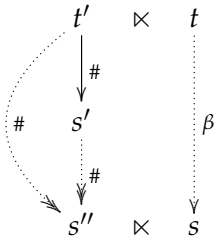
Proposition 4.5 (Reverse simulation)**.** *Let $t', s' \in \mathcal{T}^\#$ be distributive-terms and let $t \in \mathcal{T}^\lambda$ be a lambda-term such that:*

$$t \rtimes t' \to_\# s'$$

then there is a distributive term $s'' \in \mathcal{T}^\#$ and a lambda-term $s \in \mathcal{T}^\lambda$ such that:

$$t \to_\beta s \rtimes s''$$

and the step $t' \to_\# s'$ is contained in the multistep $t' \twoheadrightarrow_\# s''$. Diagramatically:



Proof. ♠ By induction on t'. The proof is constructive. □

Example 4.6. *In the example above, we needed to apply one more step in order to be able to close the diagram:*

$$\begin{array}{ccc} (\lambda^1 x.x^{[\alpha]\to\beta}[x^\alpha])[(\lambda^2 y.y^{[\alpha]\to\beta})a^{[\alpha]\to\beta}, (\lambda^3 y.y^\alpha)a^\alpha] & \ltimes & (\lambda x.xx)((\lambda y.y)a) \\ \downarrow \# & & \\ (\lambda^1 x.x^{[\alpha]\to\beta}[x^\alpha])[a^{[\alpha]\to\beta}, (\lambda^3 y.y^\alpha)a^\alpha] & & \downarrow \beta \\ \downarrow \# & & \\ (\lambda 1.xx^{[\alpha]\to\beta}[x^\alpha])[a^{[\alpha]\to\beta}, a^\alpha] & \ltimes & (\lambda x.xx)a \end{array}$$

4.3 Head normal forms

When we defined what a refinement is, we saw that some terms had refinements and others did not, but the reason behind that was not clear.

In this section we will see that terms that can be refined are exactly the terms that have a normal form. This is coherent with the fact that typability in system $\mathcal{W}$ characterizes head normalization [BKV17, Corollary 5.5].

Definition 4.7 (Head normal forms)**.** A term $t \in \mathcal{T}^\lambda$ of the lambda-calculus is a **head normal form** if it is of the form:

$$t = \lambda x_1 . \ldots \lambda x_n . y\, t_1 \ \ldots \ t_m$$

where y might be or not be among the $x_1, \ldots, x_n$. Similarly, a term $t \in \mathcal{T}^\#$ of the distributive lambda-calculus is a **head normal form** if it is of the form:

$$t = \lambda^{\ell_1} x_1 . \ldots \lambda^{\ell_n} x_n . y^\tau \vec{t}_1 \ \ldots \ \vec{t}_m$$

We say that a term *has* a head normal form if it can be reduced to a head normal form. The claim that we are putting forward is that a term can be refined if and only if it has a head normal form.

Note that for terms that *are* head normal forms this is easy. The idea is that if

$$\lambda x_1 . \ldots \lambda x_n . y\, t_1 \ \ldots \ t_m \in \mathcal{T}^\lambda$$

then it is refined by

$$\lambda^{\ell_1} x_1 . \ldots \lambda^{\ell_n} x_n . y^{[] \overset{1}{\to} \ldots \overset{n}{\to} \alpha^1} [] \ \ldots \ [] \in \mathcal{T}^\#$$

if $x_i \neq y$ for every i (otherwise the type of y is slightly different, but the term has the same shape).

The claim that terms that are refinable are exactly the ones that have a head normal form is formally stated as follows.

Lemma 4.8. *Let $t \in \mathcal{T}^\lambda$ such that it is in head normal form. Then there exists $t' \in \mathcal{T}^\#$ such that $t \rtimes t'$.*

Proof. Suppose $t \in \mathcal{T}^\lambda$ is in head normal form. Then $t = \lambda x_1 . \ldots \lambda x_n . y\, t_1 \ \ldots \ t_m$.

We will prove that there exists a correct $\Gamma \vdash t' : \sigma$ such that $t' \ltimes t$. To make the proof easier, we are going to prove something stronger: (a) that there exists such t', (b) that for every application in t' the argument list is empty, and (c) that if $x_i = y$ for any i, Γ will be empty, otherwise it will be a singleton of the form $\{y : [\tau]\}$.

We proceed by induction on the pair (n, m), that is, by induction on $\mathbb{N}^2$ with lexicographic order. ♠ □

Proposition 4.9 (Refinability characterizes head normalization)**.** *The following are equivalent:*

1. *There exists $t' \in \mathcal{T}^\#$ such that $t' \ltimes t$.*
2. *There exists $t' \in \mathcal{T}^\#$ such that $t' \ltimes t$ and $t' \twoheadrightarrow_\# \lambda^{\ell_1} x_1 . \ldots \lambda^{\ell_n} x_n . y^\tau [] \ldots []$.*

3. *There exists a head normal form s such that $t \twoheadrightarrow_\beta s$*, i.e. *$t$ has a head normal form.*

Proof. ♠ $(1 \implies 3)$ relies on Simulation (Proposition 4.3). $(2 \implies 1)$ is obvious. $(3 \implies 2)$ uses that head normal forms are refinable, plus a subject expansion lemma. Subject expansion requires formulating a strongeer property than correctness, invariant by $\to_\#$-expansion (*i.e.* by $\to_\#{}^{-1}$). □

Note that, in general, there is no need to refine a head normal form using an empty lists for all parameters. A head normal form has a corresponding Böhm tree, from which one can obtain an **approximant**, $\lambda x_1 \ldots \lambda x_n . y\, \Omega\, \Omega \ldots \Omega$ [Bar84, DCGd98]. A finite approximant is essentially any finite prefix of the Böm tree, given by the grammar $A ::= \Omega | \lambda x_1 \ldots \lambda x_n . A_1\, A_2 \ldots A_m$. $(2 \iff 3)$ from Proposition 4.9 may be generalized to arbitrary finite approximations. We did not delve into this theory.

4.4 Simulation residuals

Simulation (Proposition 4.3) ensures that every step $t \to_\beta s$ can be simulated in $\lambda^\#$ starting from a term $t' \ltimes t$. Actually, a finer relationship can be established between the derivation spaces $\mathbb{D}^\lambda(t)$ and $\mathbb{D}^\#(t')$. For this, we introduce the idea of **simulation residual**.

The input data of Proposition 4.3 consists of the term t, the step $t \to_\beta s$, and the refinement $t' \ltimes t$. Similarly as for the usual notion of residual, we define notation to denote the output data of Proposition 4.3, namely the multistep $t' \twoheadrightarrow_\# s'$, and the refinement $s' \ltimes s$.

Definition 4.10 (Simulation residuals)**.** Let $t' \ltimes t$ and let $R : t \to_\beta s$ be a step. The constructive proof of Simulation (Proposition 4.3) associates the $\to_\beta$-step R to a possibly empty set of $\to_\#$-steps $\{R_1, \ldots, R_n\}$ all of which start from t'. We write $R/t' \stackrel{\text{def}}{=} \{R_1, \ldots, R_n\}$, and we call $R_1, \ldots, R_n$ the **simulation residuals of** R **after** t'. All the complete developments of R/t' have a common target, which we denote by t'/R, called the *simulation residual of t' after R*.

Remark 4.11. If $t' \ltimes t$ and $R : t \to_\beta s$, then $t'/R \in \mathcal{T}^\#$ and $\mathsf{src}(R_i) = t'$ for all $R_i \in R/t'$.

Recall that, by abuse of notation, R/t' stands for some complete development of the set R/t'. By Simulation (Proposition 4.3), the following diagram always holds given $t' \ltimes t \to_\beta s$:

$$\begin{array}{ccc} t & \xrightarrow[R]{\beta} & s \\ \ltimes & & \ltimes \\ t' & \xrightarrow[R/t']{\#} & t'/R \end{array}$$

Example 4.12 (Examples of simulation residuals)**.** *Let $R : x\,((\lambda x.x)y) \to_\beta x\,y$.*

0. *If $t' = x^{[]\stackrel{1}{\to}\alpha^2}\,[\,]$, then $R/t' = \varnothing$.*
1. *If $t' = x^{[\alpha^1]\stackrel{2}{\to}\beta^3}\,[(\lambda^4 x.x^{\alpha^1})[y^{\alpha^1}]]$, then $R/t' = \{R'\}$, where $R' : t' \to_\# x^{[\alpha^1]\stackrel{2}{\to}\beta^3}\,[y^{\alpha^1}]$.*

2. *If* $t' = (x^{[\alpha^1,\beta^2]\xrightarrow{3}\gamma^4}[(\lambda^5 x.x^{\alpha^1})[y^{\alpha^1}],(\lambda^6 x.x^{\beta^2})[y^{\beta^2}]])$, *then* $R/t' = \{R_1, R_2\}$ *where* $R_1 : t' \to_\# t'_1$ *and* $R_2 : t' \to_\# t'_2$, *where* $t'_1 = x^{[\alpha^1,\beta^2]\xrightarrow{3}\gamma^4}[y^{\alpha^1},(\lambda^6 x.x^{\beta^2})[y^{\beta^2}]]$ *and* $t'_2 = x^{[\alpha^1,\beta^2]\xrightarrow{3}\gamma^4}[(\lambda^5 x.x^{\alpha^1})[y^{\alpha^1}],y^{\beta^2}]$. *Moreover* $t'/R = t'_3$, *where* $t'_3 = x^{[\alpha^1,\beta^2]\xrightarrow{3}\gamma^4}[y^{\alpha^1},y^{\beta^2}]$.

Simulation residuals t'/R and R/t' are defined when R is a single step. Mimicking the standard extension of residuals for derivations, we extend simulation residuals for derivations, *i.e.* t'/ρ and ρ/t' as follows.

Definition 4.13 (Simulation residuals extended to derivations)**.** If $\rho : t \twoheadrightarrow_\beta s$ is a derivation and $t' \in \mathcal{T}^\#$ is a correct term such that $t' \ltimes t$ then the **simulation residual of** t' **after** ρ is defined by induction on ρ as a term $t'/\rho \in \mathcal{T}^\#$ such that $t'/\rho \ltimes s$:

- $t'/\epsilon \stackrel{\text{def}}{=} t'$
- $t'/(R\sigma) \stackrel{\text{def}}{=} (t'/R)/\sigma$

The **simulation residual of** ρ **after** t' is defined by induction on ρ as a derivation ρ/t' such that $\rho/t' : t' \twoheadrightarrow_\# t'/\rho$:

- $\epsilon/t' \stackrel{\text{def}}{=} \epsilon$
- $(R\sigma)/t' \stackrel{\text{def}}{=} (R/t')(\sigma/(t'/R))$
  where, in the right-hand side, the expression R/t' is an abuse of notation, and it technically stands for a (canonical) complete development of the multistep R/t'.

The diagram for the inductive case is:

$$\begin{array}{ccccc} t & \xrightarrow{\quad R \quad} & \cdot & \xrightarrow{\quad \sigma \quad} & s \\ \rtimes & & \rtimes & & \rtimes \\ t' & \overset{R/t'}{\twoheadrightarrow} & t'/R & \overset{\sigma/(t'/R)}{\twoheadrightarrow} & t'/R\sigma = (t'/R)/\sigma \end{array}$$

This extension of the definition works as expected for compositions, as shown by the following lemma.

Lemma 4.14 (Simulation residuals and composition)**.** *Let* ρ, σ *be composable derivations and let* $t' \ltimes \mathsf{src}(\rho)$. *Then:*

1. $t'/\rho\sigma = (t'/\rho)/\sigma$
2. $\rho\sigma/t' = (\rho/t')(\sigma/(t'/\rho))$

Proof. Item 1. is by induction on ρ. The interesting case is when ρ is non-empty, *i.e.* $\rho = R\tau$. Then:

$$t'/R\tau\sigma = (t'/R)/\tau\sigma \stackrel{\text{h.i.}}{=} ((t'/R)/\tau)/\sigma = (t'/R\tau)/\sigma$$

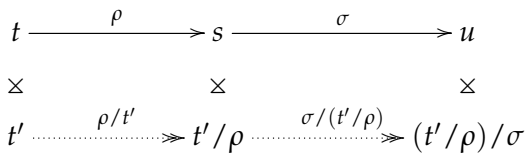
as required.

Item 2. is by induction on ρ. The interesting case is when ρ is non-empty, *i.e.* $\rho = R\tau$. Then:

$$R\tau\sigma/t' = (R/t')(\tau\sigma/(t'/R)) \stackrel{\text{h.i.}}{=} (R/t')(\tau/(t'/R))(\sigma/((t'/R)/\tau)) = (R\tau/t')(\sigma/(t'/R\tau))$$

as required. □

The result can be summarized by the following diagram.

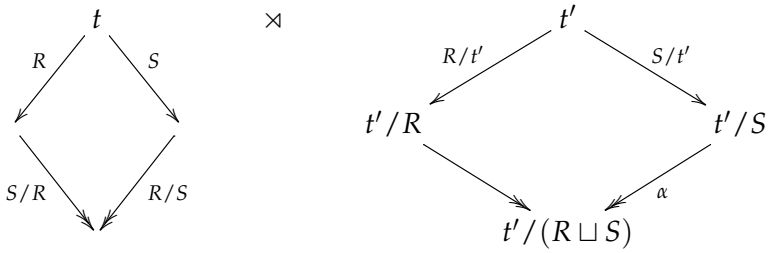


Cube lemma

The following result resembles the usual Cube Lemma[1]. It is basically a coherence result that relates different notions of residuals and checks they work together as expected.

We first enunciate a basic cube lemma that deals with single steps, which we later on will generalize to arbitrary derivations.

But before that let us explain what the cube lemma does. The situation it speaks about is as shown in the following diagram.



In the diagram before, α is a complete development of some set of steps $\mathcal{M}_\alpha$. The cube lemma reconciles the two possible ways of writing $\mathcal{M}_\alpha$. See that $\mathcal{M}_\alpha$ can be seen as projecting R/S on the term t'/S, which would suggest $\mathcal{M}_\alpha = (R/S)/(t'/S)$. On the other hand, if we only concentrate on the right hand side of the diagram, $\mathcal{M}_\alpha$ may be seen as simply the residual of R/t' after S/t', *i.e.*, $(R/t')/(S/t')$. The basic cube lemma states that this two different ways of seing $\mathcal{M}_\alpha$ yield the same set of steps.

Lemma 4.15 (Basic cube lemma for simulation residuals)**.** *Let $R : t \to s$ and $S : t \to u$ be coinitial steps, and let $t' \in \mathcal{T}^{\#}$ be a correct term such that $t' \ltimes t$. Then the following equality between sets of coinitial steps holds:*

$$(R/t')/(S/t') = (R/S)/(t'/S)$$

Observe that there are four notions of residual involved here:

$$(R \,/^{(1)}\, t') \,/^{(2)}\, (S \,/^{(1)}\, t') = (R \,/^{(3)}\, S) \,/^{(1)}\, (t' \,/^{(4)}\, S)$$

1. *Set of simulation residuals of a β-step relative to a correct term.*
2. *Set of residuals of a #-step after a #-step.*

[1] Recall that the usual Cube Lemma states that $(\rho/\sigma)/(\tau/\sigma) \equiv (\rho/\tau)/(\sigma/\tau)$; see *e.g.* [Bar84, Lemma 12.2.6].

3. *Set of residuals of a β-step after a β-step.*
4. *Simulation residual of a correct term after a β-step.*

Proof. If $R = S$ then it is easy to see that the proposition holds, so we can assume that $R \neq S$. Also, note that it is enough to see that $\mathsf{names}((R/t')/(S/t')) = \mathsf{names}(((R/S)/(t'/S))$, as we will do that in some cases. We proceed by induction on t. ♠ □

The following result relates complete developments from both worlds. It will be useful to prove the generalized version of the cube lemma, and other intermediate results.

Lemma 4.16 (Simulation residual of a development)**.** *Let $\mathcal{M}$ be a set of coinitial steps, let ρ be a complete development of $\mathcal{M}$, and let $t' \in \mathcal{T}^{\#}$ be a correct term such that $t' \ltimes \mathsf{src}(\rho)$. Then ρ/t' is a complete development of $\mathcal{M}/t'$.*

Proof. By induction on depth of $\mathcal{M}$, *i.e.* the maximum length of any development of $\mathcal{M}$.

1. $n = 0$**.** Necessarily, $\rho = \epsilon$ and $\mathcal{M} = \varnothing$. Then $\rho/t' = \epsilon$, which is a (complete) development of an empty set of steps, like $\mathcal{M}/t'$.
2. **Inductive case.** In this case, the longest development of $\mathcal{M}$ is non-trivial, so $\mathcal{M}$ must be non-empty, let it be $\{R_1, \ldots, R_{n+1}\}$ and let ρ be a complete development of it. Without loss of generality, we may assume that R_{n+1} is the first step executed by ρ, followed by a complete development of $\rho' = \{R_1/R_{n+1}, \ldots, R_n/R_{n+1}\}$, which we will call ρ'. Note that the longest development of $\mathcal{M}'$ (let's call it σ) is strictly shorter than the longest development of $\mathcal{M}$ because $R_{n+1}\sigma$ is a complete development of $\mathcal{M}$.
   By *i.h.*, $\rho'/(t'/R_{n+1})$ is a complete development of $\bigcup_{i=1}^{n} \frac{R_i/R_{n+1}}{t'/R_{n+1}}$, which by Lemma 4.15 is equal to $\bigcup_{i=1}^{n} \frac{R_i/t'}{R_{n+1}/t'}$. Then, $(R_{n+1}/t')(\rho'/(t'/R_{n+1}))$ is a complete development of $\mathcal{M}/t'$ (because by definition it is $\{R_1/t', \ldots, R_{n+1}/t'\}$).
   But note that $\rho/t' = \frac{R_{n+1}\rho'}{t'} = \frac{R_{n+1}}{t'}\frac{\rho'}{t'/R_{n+1}}$ so we are done.

□

Having shown the previous lemma, we are able to prove that the operation of projecting derivations from the pure λ-calculus to the $\lambda^{\#}$-calculus is compatible with permutation equivalence.

Proposition 4.17 (Compatibility)**.** *Let $\rho \equiv \sigma$ be permutation equivalent derivations in the λ-calculus, and let $t' \in \mathcal{T}^{\#}$ be a correct term such that $t' \ltimes \mathsf{src}(\rho)$. Then:*

1. $t'/\rho = t'/\sigma$
2. $\rho/t' \equiv \sigma/t'$

Proof. ♠ We proceed by induction on the proof that $\rho \equiv \sigma$. □

Corollary 4.18 (Simulation residuals and prefixes)**.** *If $\rho \sqsubseteq \sigma$ then $\rho/t' \sqsubseteq \sigma/t'$.*

Proof. Since $\rho \sqsubseteq \sigma$ we have that $\rho\tau \equiv \sigma$ for some derivation τ. Then by item 2 of Proposition 4.17, $\rho\tau/t' \equiv \sigma/t'$, so $(\rho/t')(\tau/(t'/\rho)) \equiv \sigma/t'$. This implies that $\rho/t' \sqsubseteq \sigma/t'$ as required. □

The cube lemma may be generalized for the case in which R and S are arbitrary derivations:

Lemma 4.19 (Generalized cube lemma for simulation residuals)**.** *Let $\rho : t \twoheadrightarrow_\beta s$ and $\sigma : t \twoheadrightarrow_\beta u$ be coinitial derivations, and let $t' \in \mathcal{T}^\#$ be a correct term such that $t' \ltimes t$. Then the following equality between derivations holds:*

$$(\rho/t')/(\sigma/t') \equiv (\rho/\sigma)/(t'/\sigma)$$

Proof. By induction on ρ using Lemma 4.15. We will prove this result in two steps. The first step will be to prove it in the case where ρ is just one step, *i.e.*, $(R/t')/(\sigma/t') \equiv (R/\sigma)/(t'/\sigma)$. We proceed by induction on σ.

1. $\sigma = \epsilon$**.** In this case $R/\epsilon = \{R\}$, $t'/\epsilon = t'$, and $\epsilon/t' = \epsilon$, so we end up with the equality we wanted, $R/t' = R/t'$.
2. $\sigma = S\sigma'$**.** First, $R/(S\sigma') = (R/S)/\sigma' = \{T/\sigma' : T \in R/S\}$. Also, $t'/\sigma = (t'/S)/\sigma'$. Then, the right hand side of the equation is $((R/S)/\sigma')/((t'/S)/\sigma')$, which in reality is the set $\{(T/\sigma')/((t'/S)/\sigma') : T \in R/S\}$, which by inductive hypothesis equals (modulo permutation equivalence) the set $\{(T/(t'/S))/(\sigma'/(t'/S)) : T \in R/S\}$, *i.e.* the set, $((R/S)/(t'/S))/(\sigma'/(t'/S))$.
   In turn, this set, by Lemma 4.15, equals the set $((R/t')/(S/t'))/(\sigma'/(t'/S))$.
   On the other hand, the left side of the equation is $(R/t')/((S\sigma')/t')$, which by definition of simulation residual equals $(R/t')/((S/t')(\sigma'/(t'/S)))$. That last derivation equals the a canonical development of the right hand side of the equation because of the general rule $\alpha/(\beta\gamma) \equiv (\alpha/\beta)/\gamma$.

Now we proceed to prove the full proposition, and we will proceed by inducion on ρ.

1. $\rho = \epsilon$**.** In this case both sides of the equation are the empty derivation.
2. $\rho = R\rho'$**.** This case can be proven with a series of equalities, in which we will abuse notation when necessary, writing R/σ to mean a complete canonical development of the actual residual set.

$$\begin{array}{rcll}
((R\rho')/\sigma)/(t'/\sigma) & = & \frac{(R\rho')/\sigma}{t'/\sigma} & \\
& = & \frac{(R/\sigma)(\rho'/(\sigma/R))}{t'/\sigma} & \text{by definition of residual} \\
& = & \frac{R/\sigma}{t'/\sigma}\frac{\rho'/(\sigma/R)}{(t'/\sigma)/(R/\sigma)} & \text{by definition of simulation residual} \\
& \equiv & \frac{R/\sigma}{t'/\sigma}\frac{\rho'/(\sigma/R)}{(t'/R)/(\sigma/R)} & \text{by the previous partial result} \\
& \equiv & \frac{R/\sigma}{t'/\sigma}\frac{\rho'/(t'/R)}{(\sigma/R)/(t'/R)} & \text{by inductive hypothesis} \\
& \equiv & \frac{R/t'}{\sigma/t'}\frac{\rho'/(t'/R)}{(\sigma/R)/(t'/R)} & \text{by the previous partial result} \\
& = & \frac{(R/t')(\rho'/(t'/R))}{\sigma/t'} & \text{by definition of residual} \\
& = & \frac{(R\rho')/t'}{\sigma/t'} & \text{by definition of simulation residual}
\end{array}$$

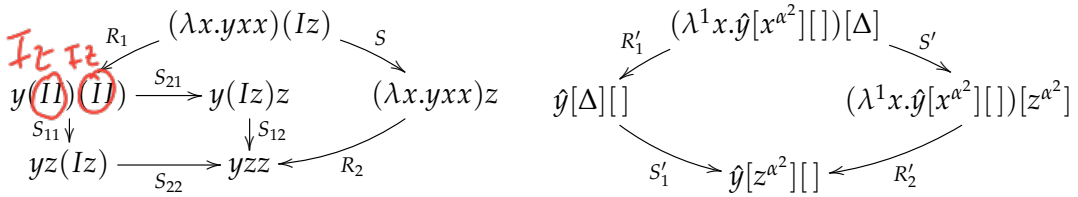
□

Corollary 4.20 (Algebraic Simulation)**.** *Let $t' \ltimes t$. Then the mapping $\mathbb{D}^\lambda(t) \to \mathbb{D}^\#(t')$ given by $[\rho] \mapsto [\rho/t']$ is a morphism of upper semilattices.*

Proof. We need to check that the morphism is monotonic and that it preserves joins. First, if $\rho \sqsubseteq \sigma$ then $\rho\tau \equiv \sigma$ for some τ. So $\rho/t \sqsubseteq (\rho/t)(\tau/(t/\rho)) = \rho\tau/t \equiv \sigma/t$ by Compatibility (Proposition 4.17).

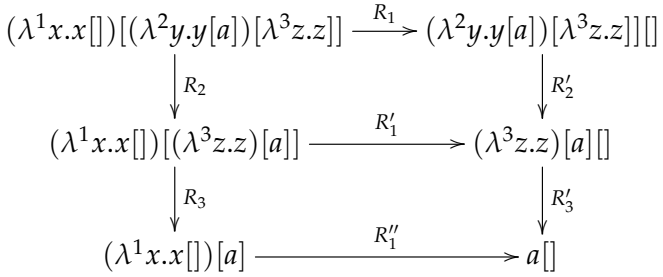
Secondly, $(\rho \sqcup \sigma)/t = \rho(\sigma/\rho)/t = (\rho/t)((\sigma/\rho)/(t/\rho)) \equiv (\rho/t)((\sigma/t)/(\sigma/\rho)) = (\rho/t) \sqcup (\sigma/t)$. □

Example 4.21. *Let $I = \lambda x.x$ and $\Delta = (\lambda^5 x.x^{\alpha^2})[z^{\alpha^2}]$ and let us write $\hat{y}$ for $y^{[\alpha^2]\xrightarrow{3}[]\xrightarrow{4}\beta^5}$. The refinement $t' := (\lambda^1 x.\hat{y}[x^{\alpha^2}][])[\Delta] \ltimes (\lambda x.yxx)(Iz)$ induces a morphism between the upper semilattices represented by the following reduction graphs:*

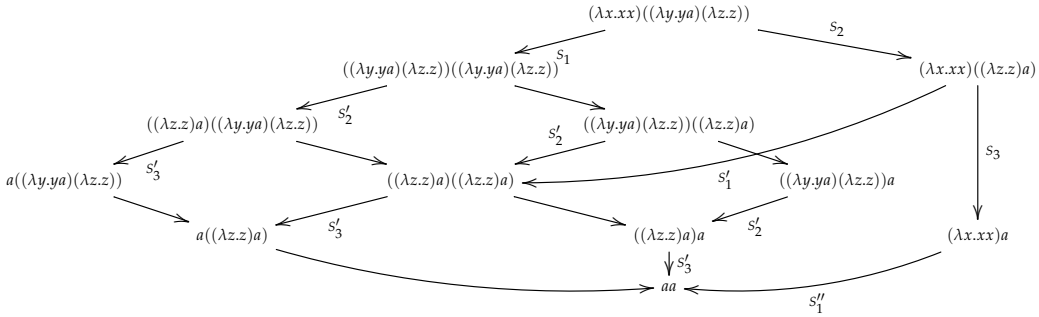


For example $(R_1 \sqcup S)/t' = (R_1 S_{11} S_{22})/t' = R'_1 S'_1 = R'_1 \sqcup S' = R_1/t' \sqcup S/t'$. Note that the step S_{22} is erased by the simulation: $S_{22}/(\hat{y}[z^{\alpha^2}][]) = \varnothing$. Intuitively, S_{22} is "garbage" with respect to the refinement $\hat{y}[z^{\alpha^2}][]$, because it lies inside an untyped *argument.*

Example 4.22. *We can do a larger example. Consider the term $(\lambda x.xx)((\lambda y.ya)(\lambda z.z))$, which is refined by $(\lambda^1 x.x[])[(\lambda^2 y.y[a])[\lambda^3 z.z]]$, among others [2]. Let's take a look at the derivation space of the distributive term. We use the labels of the steps to name the steps ($R_n, R'_n, R''_n, \ldots$ are steps that have the name n).*



Next, the derivation space of the pure lambda term. We name the steps such that a derivation below with a step S^n_i maps to the derivation with R^n_i above. The steps with no name are mapped to the empty derivation above.



Note that, for example, the derivation $S_2 S_3 S''_1$ maps to the derivation $R_2 R_3 R''_1$.

[2] The fully labeled term is $(\lambda^1 x.x^{[]\xrightarrow{5}\alpha^4}[])[(\lambda^2 y^{([]\xrightarrow{5}\alpha^4)\xrightarrow{3}([]\xrightarrow{5}\alpha^4)}.y[a^{[]\xrightarrow{5}\alpha^4}])[\lambda^3 z.z^{[]\xrightarrow{5}\alpha^4}]]$.

Notice how the unlabeled steps in the second diagram correspond to the empty derivation in the first one. That means something: the unlabeled steps are not necessary if we want to arrive to the head normal form $a\Omega$, which is "casually" the head normal form represented by $a[\,]$.

We will study this phenomenon more carefully in the next chapter.

Chapter 5

Factorization of derivations

As we observed in the two examples at the end of last chapter, when we take a derivation from the lambda calculus and project it to the distributive lambda calculus, we may end up with the empty derivation.

Remember that if we have a term $t' \in \mathcal{T}^{\#}$ that refines a $t \in \mathcal{T}^{\lambda}$, t' represents in some way a head normal form of t. Intuitively, derivations in $\mathbb{D}^{\lambda}(t)$ that map to the empty derivation in $\mathbb{D}^{\#}(t')$ will be derivations that don't do work towards carrying t to the head normal form represented by t'.

5.1 Garbage

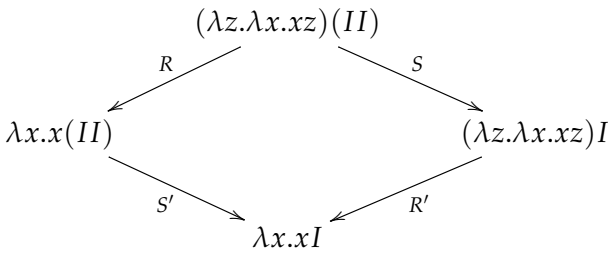
We shall call these derivations that map to the empty derivation *garbage*.

Definition 5.1 (Garbage)**.** Let $t' \ltimes t$. A derivation $\rho : t \twoheadrightarrow_\beta s$ is said to be t'**-garbage** if and only if $\rho / t' = \epsilon$.

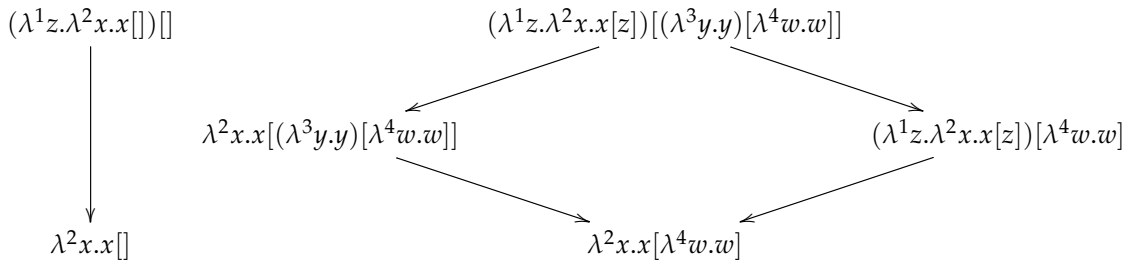
Notation 5.2. If the correct term t' is clear from the context, we will often say that ρ is **garbage**, without specifying with respect to which term. In particular, if $(\rho \cdot \sigma) : t \twoheadrightarrow_\beta s$ and $t' \ltimes t$:

- To say that ρ is garbage means that ρ is t'-garbage.
- To say that σ is garbage means that σ is (t'/ρ)-garbage.

Example 5.3. *Let $(\lambda z.\lambda x.xz)(II) \in \mathcal{T}^{\lambda}$, which has the following derivation space.*



The term above may be refined by many terms. Two of those possible refinements are $(\lambda^1 z.\lambda^2 x.x[])[]$ [1] *and* $(\lambda^1 z.\lambda^2 x.x[z])[(\lambda^3 y.y)[\lambda^4 w.w]]$ [2]*. Their derivation spaces are, respectively:*



Note that the step S is garbage with respect to the first refinement, but not with respect to the second.

For each $t' \ltimes t$, the set of t'-garbage derivations forms an *ideal* of the upper semilattice $\mathbb{D}^\lambda(t)$. More precisely:

Proposition 5.4 (Properties of garbage)**.** *Let* $t' \ltimes t$*. Then:*

1. *If* ρ *is* t'*-garbage and* $\sigma \sqsubseteq \rho$*, then* σ *is* t'*-garbage.*
2. *The composition* $\rho\sigma$ *is* t'*-garbage if and only if* ρ *is* t'*-garbage and* σ *is* (t'/ρ)*-garbage.*
3. *If* ρ *is* t'*-garbage then* ρ/σ *is* (t'/σ)*-garbage.*
4. *The join* $\rho \sqcup \sigma$ *is* t'*-garbage if and only if* ρ *and* σ *are* t'*-garbage.*

Proof. We prove each item separately.

1. Let $\sigma \sqsubseteq \rho$. Then $\sigma\tau \equiv \rho$ for some τ, so $\sigma/t' \sqsubseteq (\sigma/t')(\tau/(t'/\sigma)) = \sigma\tau/t' \equiv \rho/t'$ by Compatibility (Proposition 4.17).
2. Note that $\rho\sigma/t' = (\rho/t')(\sigma/(t'/\rho))$. So $\rho\sigma/t'$ is empty if and only if ρ/t' and $\sigma/(t'/\rho)$ are empty.
3. Suppose that $\rho/t' = \epsilon$. Then $(\rho/\sigma)/(t'/\sigma) = (\rho/t')/(\sigma/t')$ by the Cube Lemma (Lemma 4.19).
4. By the Cube Lemma (Lemma 4.19):

   $$(\rho \sqcup \sigma)/t' = \rho(\sigma/\rho)/t' = (\rho/t')((\sigma/\rho)/(t'/\rho) \equiv (\rho/t')((\sigma/t')/(\rho/t')) = (\rho/t') \sqcup (\sigma/t')$$

   So $(\rho \sqcup \sigma)/t'$ is empty if and only if ρ/t' and σ/t' are empty.

□

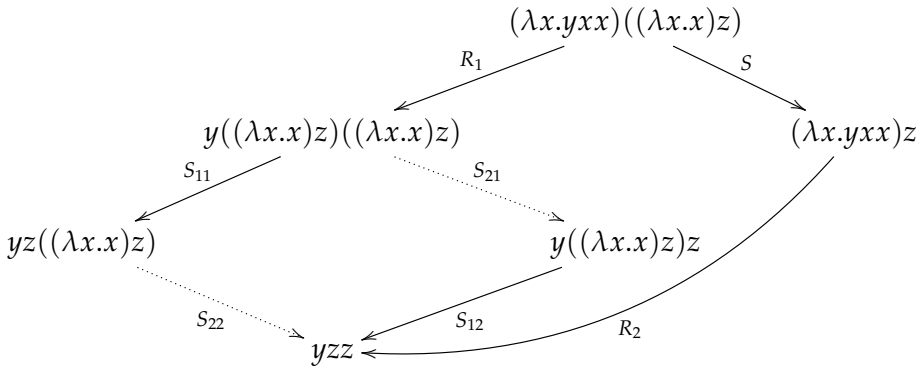
Our aim is to show that given any derivation $\rho : t \twoheadrightarrow_\beta s$ in the λ-calculus and any $t' \ltimes t$, there is a unique way of factorizing ρ as $\rho_1\rho_2$ where ρ_2 is garbage, and ρ_1 "has no garbage". The notion of garbage of Definition 5.1 is fine for our purposes. Note, in particular, that the notion of garbage is well-defined modulo permutation equivalence, *i.e.* if $\rho \equiv \sigma$ then ρ is garbage if and only σ is garbage (this is a consequence of Proposition 4.17).

However, the notion of "having no garbage" is not so easy to pin down, as seen in the following example.

[1] The fully labeled term is $(\lambda^1 z.\lambda^2 x.x^{[]\xrightarrow{3}\alpha^4}[])[]$.

[2] The fully labeled term is $(\lambda^1 z.\lambda^2 x.x^{[[\alpha^5]\xrightarrow{4}\alpha^5]\xrightarrow{2}\beta^6}[z^{[\alpha^5]\xrightarrow{4}\alpha^5}])[(\lambda^3 y.y^{[\alpha^5]\xrightarrow{4}\alpha^5})[\lambda^4 w.w^{\alpha^5}]]$.

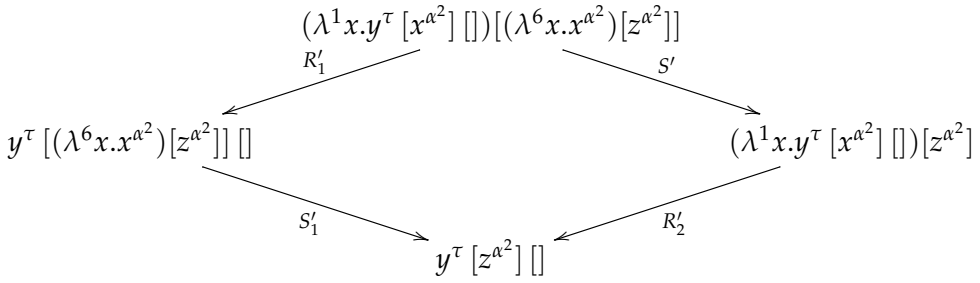
Example 5.5. *Consider the reduction space for $(\lambda x.yxx)((\lambda x.x)z)$. Dotted arrows will correspond to steps that are garbage:*



Consider moreover the following refinement:

$$(\lambda^1 x.y^{[\alpha^2]\overset{3}{\to}[]\overset{4}{\to}\beta^5}\ [x^{\alpha^2}]\ [])[(\lambda^6 x.x^{\alpha^2})[z^{\alpha^2}]] \ltimes (\lambda x.yxx)((\lambda x.x)z)$$

It has the following derivation space (letting $\tau = [\alpha^2] \overset{3}{\to} [] \overset{4}{\to} \beta^5$):



Then we can observe that:

- *$R_1, R_2, S, S_{11}, S_{12}$ are not garbage.*
- *S_{21}, S_{22} are garbage.*

In particular, the naïve *notion of "having garbage" is not well-defined modulo permutation equivalence: the derivations $R_1 S_{11} S_{22}$ and $S R_2$ are permutation equivalent, but the former contains a garbage step, while the latter does not.*

The following notion of "having no garbage" will be proven to be well-defined modulo permutation equivalence.

Definition 5.6 (Garbage-free)**.** Let $t' \ltimes t$. A derivation $\rho : t \twoheadrightarrow_\beta s$ is t'**-garbage-free** if for every derivation σ such that $\sigma \sqsubseteq \rho$:

$$\rho/\sigma \text{ is } (t'/\sigma)\text{-garbage} \quad \text{implies} \quad \rho/\sigma = \epsilon.$$

Intuitively, what the definition of garbage-free says is that we cannot strip ρ in some way such that a part is garbage: if we divide ρ in two parts σ and ρ/σ and the second part is garbage, then it actually was the empty derivation.

For example, in the situation of Example 5.5 the derivation SR_2 is **not** garbage-free, since $R_1S_{11} \sqsubseteq SR_2$ and $SR_2/R_1S_{11} \equiv S_{22}$ which is garbage but non-empty. In contrast, R_1S_{11} is garbage-free.

Lemma 5.7 (The notion of garbage-free is well-defined modulo permutation equivalence)**.** *Let $\rho : t \twoheadrightarrow_\beta s$ and $\rho \equiv \sigma$. Then ρ is t'-garbage-free if and only if σ is t'-garbage-free.*

Proof. ($\Rightarrow$) Suppose that ρ is garbage-free, and let us show that σ is garbage-free. Let $\tau \sqsubseteq \sigma$ such that σ/τ is (t'/τ)-garbage. Then $\tau \sqsubseteq \rho$ and ρ/τ is (t'/τ)-garbage. Then, since ρ is garbage-free, we have that $\rho/\tau = \epsilon$, which in turn implies that $\sigma/\tau = \epsilon$. ($\Leftarrow$) Symmetric. □

5.2 Sieving

In this subsection, we will characterize garbage-free derivations by giving a procedure–*sieving*–that, in some sense, erases all the garbage from a derivation.

Definition 5.8 (Sieving)**.** Let $t' \ltimes t$ where t' is a correct term, and let $\rho : t \twoheadrightarrow_\beta s$ be an arbitrary derivation. A step R is **coarse for** (ρ, t') if $R \sqsubseteq \rho$ and $R/t' \neq \varnothing$.

We define the **sieve of ρ with respect to t'**, written $\rho \downarrow t'$, as a derivation in the λ-calculus, going out from t, as follows, considering two cases, depending on whether there exists a coarse step for (ρ, t').

- **If there are no coarse steps for (ρ, t').** Then $(\rho \downarrow t') \stackrel{\text{def}}{=} \epsilon$.
- **If there exists a coarse step for (ρ, t').** Let R_0 the leftmost coarse step. Then:

$$(\rho \downarrow t') \stackrel{\text{def}}{=} R_0((\rho/R_0) \downarrow (t'/R_0))$$

  Note that, in the recursive invocation, the expression is well-formed because $(t'/R_0) \ltimes \mathsf{src}(\rho/R_0)$, which is immediate from Definition 4.10.

This definition is shown to be well-defined (terminating and well behaved) in the following lemmas.

Lemma 5.9 (Sieving is well-defined)**.** *The operation $\rho \downarrow t'$ is well-defined.*

Proof. ♠ We show that the recursion is well-founded using the measure $M(\rho, t') = |\rho/t'|$. □

Lemma 5.10 (Sieving is compatible with permutation equivalence)**.** *Let $\rho \equiv \sigma$. Then $\rho \downarrow t' \equiv \sigma \downarrow t'$.*

Proof. ♠ By induction on the length of $\rho \downarrow t'$, observing that $(R \sqsubseteq \rho) \iff (R/\rho = \varnothing) \iff (R/\sigma = \varnothing) \iff (R \sqsubseteq \sigma)$. □

Example 5.11. *In Example 5.3, $S \downarrow t' = S$ and $SR_2 \downarrow t' = R_1S_{11}$.*

In the next section we will prove some further results about sieving.

5.3 Some properties

What follows are two results that describe what happens when one executes a garbage step or garbage derivation. In short, if a garbage derivation creates or duplicates a step R, then R is garbage. This makes sense, as one should not need to execute garbage in order to create a "needed" step.

Lemma 5.12 (Garbage only creates garbage)**.** *Let R and S be composable steps in the λ-calculus, and let $t' \ltimes \mathsf{src}(R)$. If R creates S and R is t'-garbage, then S is (t'/R)-garbage.*

Proof. According to Lévy [Lév78], there are three creation cases in the λ-calculus. We need to consider those three cases separately. ♠ □

Lemma 5.13 (Garbage only duplicates garbage)**.** *Let R and S be coinitial steps in the λ-calculus and let $t' \ltimes \mathsf{src}(R)$. If R duplicates S,* i.e. *$\#(S/R) > 1$, and R is t'-garbage, then S is (t'/R)-garbage.*

Proof. ♠ By inspecting how R looks like given that it duplicates S. □

We now prove two propositions that give different characterizations for both garbage and garbage-free derivations. Finally, we give some general and useful properties of the sieving operation.

Proposition 5.14 (Characterization of garbage)**.** *Let $\rho : t \twoheadrightarrow_\beta s$ and $t' \ltimes t$. Then the following are equivalent:*

1. *$\rho \downarrow t' = \epsilon$.*
2. *There are no coarse steps for (ρ, t').*
3. *The derivation ρ is t'-garbage.*

Proof. It is immediate to check that items 1 and 2 are equivalent, by definition of sieving, so we focus on 2 $\implies$ 3 and 3 $\implies$ 2. ♠ □

Proposition 5.15 (Characterization of garbage-free derivations)**.** *Let $\rho : t \twoheadrightarrow_\beta s$ and $t' \ltimes t$. Then the following are equivalent:*

1. *ρ is t'-garbage-free,*
2. *$\rho \equiv \rho \downarrow t'$,*
3. *$\rho \equiv \sigma \downarrow t'$ for some derivation σ.*

Proof. We prove each direction separately. ♠ □

Proposition 5.16 (Properties of sieving)**.** *Let $t' \ltimes t$ and $\rho : t \to_\beta{}^* s$. Then:*

1. *$\rho \downarrow t'$ is t'-garbage-free and $\rho \downarrow t' \sqsubseteq \rho$.*
2. *$\rho/(\rho \downarrow t')$ is $(t'/(\rho \downarrow t'))$-garbage.*
3. *ρ is t'-garbage if and only if $\rho \downarrow t' = \epsilon$.*
4. *ρ is t'-garbage-free if and only if $\rho \downarrow t' \equiv \rho$.*

Proof. We use some technical lemmas spelled out in the appendix (♠).

1. Note that $\rho \downarrow t'$ is t'-garbage-free by Proposition 5.15. Moreover, $\rho \downarrow t' \sqsubseteq \rho$ by Lemma A.28.
2. This is precisely Lemma A.32.
3. An immediate consequence of Proposition 5.14.
4. An immediate consequence of Proposition 5.15.

□

5.4 Factorization of garbage

Notice that now, given two terms $t \ltimes t'$ and a derivation $\rho : t \to s$, we can obtain via sieving a garbage-free prefix of ρ, namely $\rho \downarrow t'$. The question is if we can somehow get a garbage derivation σ such that $(\rho \downarrow t')\sigma \equiv \rho$.

The answer is yes, as the following theorem states. We could prove it now, but in the next section we will prove a more interesting generalization of the result, that will speak not only about single derivations but of derivation spaces. When we are done with that, the following theorem will be a mere corollary.

Theorem 5.17 (Simple factorization of garbage)**.** *Let $\rho : t \twoheadrightarrow_\beta s$ and $t' \ltimes t$. Then there exist ρ_1, ρ_2 such that:*

1. *$\rho \equiv \rho_1\rho_2$*
2. *ρ_1 is t'-garbage-free,*
3. *ρ_2 is t'-garbage.*

Moreover, ρ_1 and ρ_2 are unique modulo permutation equivalence, and we have that $\rho_1 \equiv \rho \downarrow t'$ and $\rho_2 \equiv \rho/(\rho \downarrow t')$.

5.5 Lattices

As we previously said, we want to enunciate a general factorization theorem for derivation spaces. The factorization will consist in writing any derivation as the concatenation of two derivations, such that the first one is garbage-free, and the second one is garbage.

This factorization can be considered over derivation spaces because it *glues* well: the factorizations of two different derivations will be compatible in a strong sense.

Let us recall why this factorization theorem is important. If we have a term t refined by another t', then t' represents some head normal form s of t, and the fact that a derivation is garbage with respect to t' means that it does not contribute to arrive to s.

We introduce the following definitions mostly to fix nomenclature and notation:

Definition 5.18 (Upper semilattices)**.** An **upper semilattice** is a triple $(X, \leq, \vee)$ where X is a set, $\leq$ is a partial order on X, and there are binary joins $x \vee y$ for all $x, y \in X$. An **upper semilattice with bottom** is an upper semilattice with a bottom element $\bot \in X$. As customary, by abuse of notation, we write X for both the structure and the underlying set, when clear from the context. We may write $\bot_X$ to emphasize that $\bot$ is the bottom element of X.

A **morphism of upper semilattices** $f : X \to Y$ is a monotonic function $f : X \to Y$ (*i.e.* $x \leq y$ implies $f(x) \leq f(y)$) preserving joins, that is $f(x \vee y) = f(x) \vee f(y)$. A **morphism of upper semilattices with bottom** moreover preserves the bottom element, *i.e.* $f(\bot) = \bot$.

Upper semilattices provided with morphisms form a category USL. Upper semilattices with bottom provided with morphisms form a category USLB. Any upper semilattice may be regarded as a category whose objects are the elements of X, and such

that there is a (unique) morphism $x \to y$ if and only if $x \leq y$. We write $x \hookrightarrow y$ for such morphism.

As usual, if X and Y are posets, the set of functions $f : X \to Y$ is also a poset with $f \leq g$ defined as $f(x) \leq g(x)$ for all $x \in X$. Thus USL forms a 2-category in which 0-cells are upper semilattices, 1-cells are morphisms of upper semilattices $f : X \to Y$, and there is a 2-cell $f \Rightarrow g$ if and only if $f \leq g$.

Definition 5.19 (Lax 2-functor)**.** Let $\mathscr{C}$ be a category and $\mathscr{D}$ a 2-category. A lax 2-functor $F : \mathscr{C} \to \mathscr{D}$ is a pair (F_1, F_2) where $F_1 : \mathsf{Ob}(\mathscr{C}) \to \mathsf{Ob}(\mathscr{D})$ is a function, and for every morphism $f : X \to Y$ in $\mathscr{C}$,

$$F_2(f) : F_1(X) \to F_1(Y) \qquad \text{is a 1-cell in } \mathscr{D}$$

such that functor laws hold up to 2-cells. We will be interested in the following notion of lax 2-functor:

1. $F_2(\mathsf{id}_X) = \mathsf{id}_{F_1(X)}$ for all $X \in \mathsf{Ob}(\mathscr{C})$.
2. $F_2(f \circ g) \leq F_2(f) \circ F_2(g)$ is a 2-cell for any two morphisms $g : X \to Y$, $f : Y \to Z$ in $\mathscr{C}$.

As usual with functors, we may write F to stand for either F_1 or F_2, when clear from the context. Related lax functor definitions can be found in [Str72].

Recall that we will have a factorization for each derivation, and we want to somehow glue those factorizations such that factorizations behave well under semilattice operations. That can be done with an adaptation of Grothendieck's construction for partially ordered sets.

The Grothendieck construction allows us to perform a "twisted product" between a structure and a functor from that structure. This is what we want to do because we have a structure that represents garbage-free derivations from a term, and we have a functor from that structure to the category of semi-lattices: for every garbage-free derivation ρ we have a semi-lattice that represents the garbage derivations $\{\sigma \mid \mathsf{src}(\sigma) = \mathsf{tgt}(\rho)\}$.

Definition 5.20 (Grothendieck construction for partially ordered sets)**.** Let A be a poset, and let $B : A \to \mathsf{Poset}$ be a mapping associating each object $a \in A$ to a poset $B(a)$. Suppose moreover that B is a *lax 2-functor*. More precisely, for each $a \leq b$ in A let $B(a \hookrightarrow b) : B(a) \to B(b)$ be a monotonic function such that:

1. $B(a \hookrightarrow a) = \mathsf{id}_a$ for all $a \in A$.
2. $B((b \hookrightarrow c) \circ (a \hookrightarrow b)) \leq B(b \hookrightarrow c) \circ B(a \hookrightarrow b)$ is a 2-cell for all $a \leq b \leq c$ in A.

The *Grothendieck construction* $\int_A B$ is defined as the poset given by $\{(a, b) \mid a \in A,\ b \in B(a)\}$ and such that $(a, b) \leq (a', b') \overset{\text{def}}{\iff} a \leq a'$ and $B(a \hookrightarrow a')(b) \leq b'$.

It is routine to check that $\int_A B$ is indeed a poset.

Proposition 5.21 (Semilattices of garbage-free and garbage derivations)**.** *Let $t' \ltimes t$. Then:*

1. *The set $\mathbb{F}(t', t) = \{[\rho] \mid \mathsf{src}(\rho) = t$ and ρ is t'-garbage-free$\}$ is a finite lattice, with the order $[\rho] \trianglelefteq [\sigma] \overset{\text{def}}{\iff} \rho/\sigma$ is (t'/σ)-garbage, the join $[\rho] \triangledown [\sigma] = [(\rho \sqcup \sigma) \downarrow t']$, and the meet $[\rho] \vartriangle [\sigma]$ given by the join of all the $[\tau]$ such that $[\tau] \trianglelefteq [\rho]$ and $[\tau] \trianglelefteq [\sigma]$.*

2. *The set* $\mathbb{G}(t', t) = \{[\rho] \mid \mathsf{src}(\rho) = t \text{ and } \rho \text{ is } t'\text{-garbage}\}$ *is an upper semilattice, with the structure inherited from* $\mathbb{D}^\lambda(t)$.

Proof. ♠ The proof relies on the properties of garbage (Proposition 5.4) and sieving (Proposition 5.16). □

Suppose that $t' \ltimes t$, and let $\mathcal{F} \stackrel{\text{def}}{=} \mathbb{F}(t', t)$ denote the lattice of t'-garbage-free derivations. Let $\mathcal{G} : \mathcal{F} \to \mathsf{Poset}$ be the lax 2-functor $\mathcal{G}([\rho]) \stackrel{\text{def}}{=} \mathbb{G}(t'/\rho, \mathsf{tgt}(\rho))$ with the action on morphisms $\mathcal{G}([\rho] \hookrightarrow_{\mathcal{F}} [\sigma]) : \mathcal{G}([\rho]) \to \mathcal{G}([\sigma])$ given by $[\alpha] \mapsto [\rho\alpha/\sigma]$. Then:

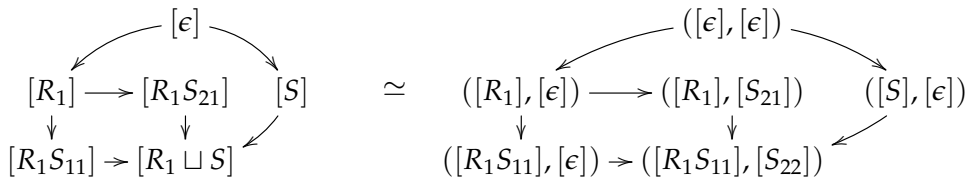
Theorem 5.22 (Factorization)**.** *The Grothendieck construction* $\int_{\mathcal{F}} \mathcal{G}$ *is an upper semilattice. The join is given by* $(a, b) \vee (a', b') = (a \triangledown a', \mathcal{G}(a \hookrightarrow_{\mathcal{F}} a \triangledown a')(b) \sqcup \mathcal{G}(a' \hookrightarrow_{\mathcal{F}} a \triangledown a')(b'))$. *Moreover, the following is an isomorphism of upper semilattices:*

$$\begin{array}{rcl} \mathbb{D}^\lambda(t) & \to & \int_{\mathcal{F}} \mathcal{G} \\ [\rho] & \mapsto & ([\rho \downarrow t'], [\rho/(\rho \downarrow t')]) \end{array} \qquad \begin{array}{rcl} \int_{\mathcal{F}} \mathcal{G} & \to & \mathbb{D}^\lambda(t) \\ ([\rho], [\sigma]) & \mapsto & [\rho\sigma] \end{array}$$

Proof. ♠ The proof consists in verifying that $\mathcal{G}$ is a lax 2-functor, then that $\int_{\mathcal{F}} \mathcal{G}$ is an upper semilattice, and finally that the mappings above are an isomorphism. □

As an immediate consequence of this theorem, any derivation ρ in the λ-calculus may be decomposed as $\rho \equiv \rho_1\rho_2$, where ρ_1 is t'-garbage-free and ρ_2 is (t'/ρ_1)-garbage. Moreover, ρ_1 and ρ_2 are unique, modulo permutation equivalence. Note that this is exactly what we stated in the factorization theorem in last section, Theorem 5.17.

Example 5.23. *Let* $t = (\lambda x.yxx)(Iz)$ *and* t' *be as in Example 4.21. The upper semilattice* $\mathbb{D}^\lambda(t)$ *can be factorized as* $\int_{\mathcal{F}} \mathcal{G}$ *as follows. Here posets are represented by their Hasse diagrams:*



Note for example that $([S], [\epsilon]) \leq ([R_1S_{11}], [S_{22}])$ *because* $[S] \trianglelefteq [R_1S_{11}]$, *that is,* $S/R_1S_{11} = S_{22}$ *is garbage, and* $\mathcal{G}([S] \hookrightarrow_{\mathcal{F}} [R_1S_{11}])([\epsilon]) = [S/R_1S_{11}] = [S_{22}] \sqsubseteq [S_{22}]$.

Chapter 6

Conclusions

In this thesis we defined a calculus based on non-idempotent intersection types, which we called $\lambda^{\#}$. Admittedly, its syntax is complex because of the labeling of variables and the correctness invariant, which where *ad hoc* additions so that the calculus was confluent. However the derivation spaces of this calculus are very simple structures, they are distributive lattices (Theorem 3.27), and are representable as rings of sets (Theorem 3.29),

Then we proved that derivation spaces in the λ-calculus can be mapped onto these simpler derivation spaces, via a strong simulation result (Corollary 4.20). Using this, we proved how the derivation space of any λ-term that has a head normal form can be factorized as a "twisted product" of garbage-free and garbage derivations (Theorem 5.22).

We think this validates the hypothesis that explicitly representing resource management can shed some light on the structure of derivation spaces. We would like to know what would happen if we changed $\lambda^{\#}$ for another resource calculus, whether or not similar results can be found.

The Factorization theorem (Theorem 5.22) is reminiscent of Melliès' [Mel97] external–internal factorization. It should be possible to establish a formal correspondence between these notions. As noted by Melliès, any evaluation strategy that always picks external steps is hypernormalizing. It should be easy to show that this holds for evaluation strategies picking non-garbage steps, using the terminology of this paper.

Open is the question that we posed after Proposition 4.9, to study the relationship between refinements of a term of the λ-calculus and its approximants. It should not be hard to prove that there is a correspondence, *i.e.* that for a given term t and an approximant A, we can find one refinement of t whose normal form refines a head normal form of t that corresponds to A.

A related question is the one of whether it is possible to characterize garbage in the same way we did in this work without using the $\lambda^{\#}$-calculus. We believe a plausible way to do this would be to use approximants: instead of defining garbage with respect to a refinement we define garbage with respect to an approximant. Such a result would not have any $\lambda^{\#}$ traces in its statement, but $\lambda^{\#}$ may help to prove it easily.

Appendix A

Appendix

A.1 Proof of Lemma 2.8 — Unique typing

We will prove that given two typings of a term, they are the same.

Proof. By induction on t.

1. **Variable,** $t = x^\sigma$**.** If $\Gamma \vdash x^\sigma : \tau$, then the last rule in the derivation must be `var` so it must be the case that $\tau = \sigma$ and $\Gamma = \{x : [\tau]\}$. The same is true for τ' and Γ'. Note that the derivations are equal.
2. **Abstraction,** $t = \lambda y.u$**.** As $\Gamma \vdash \lambda y.u : \tau$, then for some $[\rho_1, \ldots, \rho_n]$ and τ_1 it must be the case that $\tau = [\rho_1, \ldots, \rho_n] \xrightarrow{\ell} u_1$. Given that the last rule in the derivation must be $\to_{\mathrm{I}}$, we know that
$$\frac{\Gamma \oplus y : [\rho_1, \ldots, \rho_n] \vdash u : \tau_1}{\Gamma \vdash \lambda^\ell y.u : [\rho_1, \ldots, \rho_n] \xrightarrow{\ell} \tau_1} \to_{\mathrm{I}}$$
The same way, there are $[\rho'_1, \ldots, \rho'_n]$ and τ'_1 such that
$$\Gamma' \oplus y : [\rho'_1, \ldots, \rho'_n] \vdash u : \tau'_1$$
But now we can use our inductive hypothesis, that tells us that $\Gamma \oplus y : [\rho_1, \ldots, \rho_n] = \Gamma' \oplus y : [\rho'_1, \ldots, \rho'_n]$ and $\tau_1 = \tau'_1$, which implies that $\Gamma = \Gamma'$ and $\tau = \tau'$. Moreover, the inductive hypothesis tells us that the derivations are the same.
3. **Application,** $t = u[r_1, \ldots, r_n]$**.** Given that the last rule in the derivation must be $\to_{\mathrm{E}}$, we know that
$$\frac{\Gamma_0 \vdash u : [\sigma_1, \ldots, \sigma_n] \xrightarrow{\ell} \tau \qquad (\Gamma_i \vdash r_i : \sigma_i)_{i=1}^n}{\Gamma = \Gamma_0 +_{i=1}^n \Gamma_i \vdash u[r_1, \ldots, r_n] : \tau} \to_{\mathrm{E}}$$
The same way,
$$\frac{\Gamma'_0 \vdash u : [\sigma'_1, \ldots, \sigma'_n] \xrightarrow{\ell'} \tau' \qquad (\Gamma'_i \vdash r_i : \sigma'_i)_{i=1}^n}{\Gamma' = \Gamma'_0 +_{i=1}^n \Gamma'_i \vdash u[r_1, \ldots, r_n] : \tau'} \to_{\mathrm{E}}$$
Using the inductive hypothesis for u, we get that $\Gamma_0 = \Gamma'_0$ and $[\sigma_i]_{i=1}^n \xrightarrow{\ell} \tau = [\sigma'_j]_{j=1}^n \xrightarrow{\ell'} \tau'$. In particular, $\tau = \tau'$. The inductive hypothesis also tells us that the derivations are the same.

Lastly, using the inductive hypothesis n times for each r_i, we get that $\Gamma_i = \Gamma'_i$ (given that the derivations are the same), so adding everything we can see that $\Gamma_0 +_{i=1}^{n} \Gamma_i = \Gamma'_0 +_{i=1}^{n} \Gamma'_i$, i.e. $\Gamma = \Gamma'$ (and more generally that the derivations are the same).

□

A.2 Proof of Lemma 2.17 — Linearity

Let $t \in \mathcal{T}^{\#}$ be a correct term, and $\Gamma \vdash t : \tau$ its (unique) type derivation. Let x be any variable, and consider the $n \geq 0$ free occurrences of the variable x in the term t, more precisely, write t as $t = \hat{C}\langle x^{\tau_1}, \dots, x^{\tau_n}\rangle$, where $\hat{C}$ is a context with n-holes such that $x \notin \mathsf{fv}(\hat{C})$. We will prove that $\Gamma(x) = \{\tau_1, \dots, \tau_n\}$.

Proof. By induction on t.

1. **Variable (same),** $t = x^{\tau}$**.** By uniqueness of derivations, we have that $\{x : \{\tau\}\} \vdash x^{\tau} : \tau$. Also, $\hat{C} = \Box$ and $t = \hat{C}\langle x^{\sigma}\rangle$. And we also have that $\Gamma(x) = \{\tau\}$, so we are done.
2. **Variable (different),** $t = y^{\tau}$**.** By uniqueness of derivations, we have that $\{y : \{\tau\}\} \vdash y^{\tau} : \tau$. Also, $\hat{C} = y^{\tau}$ (that is, $\hat{C}$ does not have any holes) and $t = \hat{C}$. And we also have that $\Gamma(x) = \varnothing$, so we are done.
3. **Abstraction,** $t = \lambda y.u$**.** By uniqueness of derivations, we have that the derivation of t is the following.

$$\frac{\Gamma \oplus y : [\rho_1, \dots, \rho_n] \vdash u : \tau_1}{\Gamma \vdash \lambda^{\ell} y.u : [\rho_1, \dots, \rho_n] \xrightarrow{\ell} \tau_1} \to_{\mathrm{I}}$$

   Given that u is a subterm of t, it is correct. So we may use the inductive hypothesis, which tells us that there is a context $\hat{C}_1$ such that $u = \hat{C}_1\langle x^{\tau_1}, \dots, x^{\tau_n}\rangle$ (where $x \notin \mathsf{fv}(\hat{C})$) and $(\Gamma \oplus y : [\rho_1, \dots, \rho_n])(x) = \{\tau_1, \dots, \tau_n\}$.
   Then, we take $\hat{C}$ to be $\lambda^{\ell} y.\hat{C}_1$. Note that $x \notin \mathsf{fv}(\hat{C})$, and also that $\Gamma(x) = (\Gamma \oplus y : [\rho_1, \dots, \rho_n])(x) = \{\tau_1, \dots, \tau_n\}$.
4. **Application,** $t = u[r_1, \dots, r_n]$**.** By uniqueness of derivations, we have that the derivation of t is the following.

$$\frac{\Gamma_0 \vdash u : [\sigma_1, \dots, \sigma_n] \xrightarrow{\ell} \tau \qquad (\Gamma_i \vdash r_i : \sigma_i)_{i=1}^{n}}{\Gamma = \Gamma_0 +_{i=1}^{n} \Gamma_i \vdash u[r_1, \dots, r_n] : \tau} \to_{\mathrm{E}}$$

   Given that u and all r_i are subterms of t, they are correct; so can use inductive hypothesis on them. By inductive hypothesis on u, we get that $u = \hat{C}_0\langle x^{\tau_{0,0}}, \dots, x^{\tau_{0,m_0}}\rangle$ (where $x \notin \mathsf{fv}(\hat{C}_0)$) and $\Gamma_0(x) = \{\tau_{0,0}, \dots, \tau_{0,m_0}\}$. On the other hand, by inductive hypothesis on r_i, we get that $r_i = \hat{C}_i\langle x^{\tau_{i,0}}, \dots, x^{\tau_{i,m_i}}\rangle$ (where $x \notin \mathsf{fv}(\hat{C}_i)$) and $\Gamma_i(x) = \{\tau_{i,0}, \dots, \tau_{i,m_i}\}$.
   If we take $\hat{C}$ to be $\hat{C}_0[\hat{C}_1, \dots, \hat{C}_n]$, we have that $x \notin \mathsf{fv}(\hat{C})$ and

$$\begin{aligned} t &= u[r_1, \dots, r_n] \\ &= \hat{C}_0\langle x^{\tau_{0,0}}, \dots, x^{\tau_{0,n_0}}\rangle[\hat{C}_1\langle x^{\tau_{1,0}}, \dots, x^{\tau_{1,m_1}}\rangle, \dots \hat{C}_n\langle x^{\tau_{n,0}}, \dots, x^{\tau_{n,m_n}}\rangle] \\ &= (\hat{C}_0[\hat{C}_1, \dots, \hat{C}_n])\langle x^{\tau_{0,0}}, \dots, x^{\tau_{0,n_0}}, \dots, x^{\tau_{n,0}}, \dots, x^{\tau_{n,m_n}}\rangle \end{aligned}$$

To end the proof, we check the condition on the context.

$$\begin{aligned}\Gamma(x) &= (\Gamma_0 +_{i=1}^{n} \Gamma_i)(x)\\ &= \Gamma_0(x) +_{i=1}^{n} \Gamma_i(x)\\ &= \{\tau_{0,0}, \ldots, \tau_{0,m_0}\} +_{i=1}^{n} \{\tau_{i,0}, \ldots, \tau_{i,m_i}\}\\ &= \{\tau_{0,0}, \ldots, \tau_{0,m_0}, \ldots, \tau_{n,0}, \ldots, \tau_{n,m_n}\}\end{aligned}$$

□

A.3 Proof of Lemma 2.19 — $\mathcal{T}^{\#}$ is closed under $\to_{\#}$

We are to prove that if t is correct and $t \to_{\#} t'$, then t' is correct. We need a few auxiliary results:

Lemma A.1 (Subterm property)**.** *Let $t \in \mathcal{T}^{\#}$. Then every subterm of t is in $\mathcal{T}^{\#}$.*

Proof. Straightforward by induction on t. □

Lemma A.2. *Let $t \in \mathcal{T}^{\#}$ such that $t \to_{\#} t'$. Then the set of labels that decorate the lambdas in t' is strictly contained in that of t.*

Proof. We proceed by induction on t. Given a term s, we write $L(s)$ for the set of labels of the lambdas of s.

1. **Variable $t = x$.** This case is impossible given that there are no redexes.
2. **Abstraction, $t = \lambda^{\ell} x.s$.** In this case, $t' = \lambda x.s'$, where $s \to_{\#} s'$. By inductive hypothesis, $L(s') \subset L(s)$. Then, $L(t') = \{\ell\} \cup L(s') \subset \{\ell\} \cup L(s) = L(t)$.
3. **Application, $t = r[u_1, \ldots, u_n]$.** We have three subcases, depending on the place where reduction takes place. It can be either in r, in one of the u_i or at the root.
   - 3.1 **Reduction is internal to r.** Then $t' = r'[u_1, \ldots, u_n]$, where $r \to_{\#} r'$. By indutive hypothesis, $L(r') \subset L(r)$. Then, $L(t') = L(r') \cup_i L(u_i) \subset L(r) \cup_i L(u_i) = L(t)$.
   - 3.2 **Reduction is internal to u_k.** Then $t' = r[u_1, \ldots, u'_k, \ldots, u_n]$, where $u_k \to_{\#} u'_k$. By indutive hypothesis, $L(u'_k) \subset L(u_k)$. Then, $L(t') = L(r') \cup_{i \neq k} L(u_i) \cup L(u'_k) \subset L(r') \cup_{i \neq k} L(u_i) \cup L(u_k) = L(r) \cup_i L(u_i) = L(t)$.
   - 3.3 **Reduction is at the root.** In this case, $r = \lambda^{\ell} x.r'$, so $t' = r'\{x := [u_1, \ldots, u_n]\}$. Given that $L(t) = \{e\} \cup L(r') \cup_i L(u_i)$, it would be enough to show that $L(t') = L(r') \cup_i L(u_i)$. We will prove that $L(r'\{x := [u_1, \ldots, u_n]\}) = L(r') \cup_i L(u_i)$ by induction on r'.
     - 3.3.1 **Variable (same), $r' = x$.** In this case $n = 1$, $L(r') = \varnothing$ and $L(x\{x := [u_1]\}) = L(u_1)$.
     - 3.3.2 **Variable (different), $r' = y$.** In this case $n = 0$ and $L(y\{x := [\,]\}) = L(y) = [\,]$.
     - 3.3.3 **Abstraction, $r' = \lambda^{\ell} y.r''$.**

$$\begin{aligned}L((\lambda^{\ell} y.r'')\{x := [u_1, \ldots, u_n]\}) &= L(\lambda^{\ell} y.r''\{x := [u_1, \ldots, u_n]\})\\ &= \{\ell\} \cup L(r''\{x := [u_1, \ldots, u_n]\})\\ &\overset{\text{h.i.}}{=} \{\ell\} \cup L(r'') \cup_i L(u_i)\\ &= L(\lambda^{\ell} y.r'') \cup_i L(u_i)\\ &= L(r') \cup_i L(u_i)\end{aligned}$$

3.3.4 **Application,** $r' = r''\vec{r''}$**.**

$$\begin{aligned} L((r''\vec{r''})\{x := \vec{u}\}) &= L(r''\{x := \vec{u}_0\}[r_i''\{x := \vec{u}_i\}]_{i=1}^m) \\ &= L(r''\{x := \vec{u}_0\}) \cup_i L(r_i''\{x := \vec{u}_i\}) \\ &\overset{\text{h.i.}}{=} \left(L(r'') \cup_j L(u_{0j})\right) \cup_i \left(L(r_i'') \cup_j L(u_{ij})\right) \\ &= L(r'') \cup_i L(r_i'') \cup_k L(u_k) \\ &= L(r''\vec{r''}) \cup_k L(u_k) \\ &= L(r') \cup_k L(u_k) \end{aligned}$$

□

Now we proceed to prove the main result.

$\mathcal{T}^\#$ is closed under $\to_\#$

Proof. Let $t = \mathtt{C}\langle(\lambda^\ell x.s)\vec{u}\rangle$. We proceed by induction on $\mathtt{C}$.

1. **Empty context,** $\mathtt{C} = \square$**.** Then $t = (\lambda^\ell x.s)\vec{u}$ and $t' = s\{x := \vec{u}\}$. The terms s and u_i are correct by Lemma A.1, and also have pairwise distinct labels in their lambdas. Let's prove that if $\Gamma \vdash s : \sigma$ and $\Delta_i \vdash u_i : \tau_i$ are correct, σ is sequential, $\Gamma +_{i=1}^n \Delta$ is sequential and s and u have pairwise distinct labels in their lambdas then $s\{x := \vec{u}\}$ also is correct. We proceed by induction on s.
   - 1.1 **Variable (same).** $x^\tau\{x := [u]\} = u$, which is correct by hypothesis.
   - 1.2 **Variable (different).** $y^\tau\{x := []\} = y$, which is correct by hypothesis.
   - 1.3 **Abstraction.** $(\lambda^\ell y.r)\{x := \vec{u}\} = \lambda^\ell y.r\{x := \vec{u}\}$. All u_i are correct by hypothesis, and r is correct by Lemma A.1. Then by inductive hypothesis $r\{x := \vec{u}\}$ is correct.
     Also, it is the case that $\dfrac{\Gamma \oplus y : \mathcal{M} \vdash r\{x := \vec{u}\} : \sigma}{\Gamma \vdash \lambda^\ell y.r\{x := \vec{u}\} : \mathcal{M} \xrightarrow{\ell} \sigma} \to_{\mathrm{I}}$
     - **Unique lambdas.** As $(\lambda^\ell y.r)$ is correct and it has pairwise distinct lambda labels with u_i, then ℓ does not decorate any lambda in r or u_i. Then, as $r\{x := \vec{u}\}$ has unique lambdas, $\lambda^\ell y.r\{x := \vec{u}\}$ also has.
     - **Sequential contexts.** The derivation of $r\{x := \vec{u}\}$ has sequential contexts because it is correct. Also, Γ is sequential because $\Gamma \oplus y : \mathcal{M}$ is sequential.
     - **Sequential types.** Because it is correct, all types in the derivation of $r\{x := \vec{u}\}$ are sequential. Finally, $\mathcal{M}$ is sequential because it is $(\Gamma \oplus y : \mathcal{M})(y)$, which is sequential.
   - 1.4 **Application.** $(r\vec{r})\{x := \vec{u}\} = r\{x := \vec{u}_0\}[r_1\{x := \vec{u}_1\}, \ldots, r_n\{x := \vec{u}_n\}]$. By inductive hypothesis, $r\{x := \vec{u}_0\}$ and all $r_i\{x := \vec{u}_i\}$ are correct.
     By $\to_{\mathrm{E}}$, $\dfrac{\Gamma_0 \vdash r : [\sigma_1, \ldots, \sigma_n] \xrightarrow{\ell} \tau \qquad (\Gamma_i \vdash r_i : \sigma_i)_{i=1}^n}{\Gamma = \Gamma_0 +_{i=1}^n \Gamma_i \vdash r\vec{r} : \tau} \to_{\mathrm{E}}$
     And $\Delta_i \vdash u_i : \sigma_i$ for all $i \in \{1, \ldots, n\}$.
     By $\to_{\mathrm{E}}$,

$$\frac{\Gamma_0 + \vec{\Delta}_0 \vdash r\{x := \vec{u}_0\} : [\sigma_1, \ldots, \sigma_n] \xrightarrow{\ell} \tau \qquad \left(\Gamma_i + \vec{\Delta}_i \vdash s_i : r_i\{x := \vec{u}_i\}\right)_{i=1}^n}{\Gamma_0 +_{i=1}^n \Gamma_i +_{i=0}^n \Delta_i \vdash r\{x := \vec{u}_0\}[r_1\{x := \vec{u}_1\}, \ldots, r_n\{x := \vec{u}_n\}] : \tau} \to_{\mathrm{E}}$$

- **Unique lambdas.** By hypothesis r, $\vec{r}$ and $\vec{u}$ all have pairwise distinct labels in their lambdas.
- **Sequential contexts.** The derivations of $r\{x := \vec{u}_0\}$ and $r_i\{x := \vec{u}_i\}$ have sequential contexts because the terms are correct. Also, $\Gamma_0 +_{i=1}^{n} \Gamma_i +_{i=0}^{n} \Delta_i$ is correct by hypothesis.
- **Sequential types.** Because they are correct, all types in the derivations of $r\{x := \vec{u}_0\}$ and $r_i\{x := \vec{u}_i\}$ are sequential. Finally, if τ is an arrow type, its domain is sequential by hypothesis.

2. **Under an abstraction,** $\mathtt{C} = \lambda^{\ell'} y.\mathtt{C}'$**.** Then $t = \lambda^{\ell'} y.\mathtt{C}'\langle(\lambda^{\ell} x.s)\vec{u}\rangle$ and $t' = \lambda^{\ell'} y.\mathtt{C}'\langle s\{x := \vec{u}\}\rangle$. By Lemma A.1, $\mathtt{C}'\langle(\lambda^{\ell} x.s)\vec{u}\rangle$ is correct, so by i.h. $\mathtt{C}'\langle s\{x := \vec{u}\}\rangle$ is correct.
   Let $\Gamma \vdash \lambda^{\ell'} y.\mathtt{C}'\langle(\lambda^{\ell} x.s)\vec{u}\rangle : \mathcal{M} \xrightarrow{\ell'} \sigma$.
   By Subject reduction, $\dfrac{\Gamma \oplus y : \mathcal{M} \vdash \mathtt{C}'\langle s\{x := \vec{u}\}\rangle : \sigma}{\Gamma \vdash \lambda^{\ell'} y.\mathtt{C}'\langle s\{x := \vec{u}\}\rangle : \mathcal{M} \xrightarrow{\ell'} \sigma} \to_{\mathrm{I}}$
   Then $\lambda^{\ell'} y.\mathtt{C}'\langle s\{x := \vec{u}\}\rangle$ is correct, because
   - **Unique lambdas.** As it is correct, $\mathtt{C}'\langle s\{x := \vec{u}\}\rangle$ enjoys uniqueness of lambda labeling. Also, as t as correct, ℓ' does not decorate any lambda in $\mathtt{C}'\langle(\lambda^{\ell} x.s)\vec{u}\rangle$ so by Lemma A.2, ℓ' does not decorate any lambda in $\mathtt{C}'\langle s\{x := \vec{u}\}\rangle$.
   - **Sequential contexts.** The derivation of $\mathtt{C}'\langle s\{x := \vec{u}\}\rangle$ has sequential contexts because it is correct. Also, Γ is sequential because t is correct.
   - **Sequential types.** Because it is correct, all types in the derivation of $\mathtt{C}'\langle s\{x := \vec{u}\}\rangle$ are sequential. Finally, $\mathcal{M}$ is sequential because t is correct.

3. **Left of an application,** $\mathtt{C} = \mathtt{C}'\,\vec{r}$**.** Then $t = \mathtt{C}'\langle(\lambda^{\ell} x.s)\vec{u}\rangle\,\vec{r}$ and $t' = s\{x := \vec{u}\}\,\vec{r}$. By Lemma A.1, $\mathtt{C}'\langle(\lambda^{\ell} x.s)\vec{u}\rangle$ is correct, so by i.h. $\mathtt{C}'\langle s\{x := \vec{u}\}\rangle$ is correct.
   Let $\Gamma \vdash \mathtt{C}'\langle(\lambda^{\ell} x.s)\vec{u}\rangle\,\vec{r} : \tau$.
   By Subject reduction, $\dfrac{\Gamma_0 \vdash \mathtt{C}'\langle s\{x := \vec{u}\}\rangle : [\sigma_1, \ldots, \sigma_n] \xrightarrow{\ell} \tau \qquad (\Gamma_i \vdash r_i : \sigma_i)_{i=1}^{n}}{\Gamma = \Gamma_0 +_{i=1}^{n} \Gamma_i \vdash \mathtt{C}'\langle s\{x := \vec{u}\}\rangle\,[r_1, \ldots, r_n] : \tau} \to_{\mathrm{E}}$
   Then $\mathtt{C}'\langle s\{x := \vec{u}\}\rangle\,\vec{r}$ is correct, because
   - **Unique lambdas.** As it is correct, $\mathtt{C}'\langle s\{x := \vec{u}\}\rangle$ enjoys uniqueness of lambda labeling. Also, as t as correct, the labels that decorate all the r_i do not decorate any lambda in $\mathtt{C}'\langle(\lambda^{\ell} x.s)\vec{u}\rangle$ so by Lemma A.2, they do not decorate any lambda in $\mathtt{C}'\langle s\{x := \vec{u}\}\rangle$. r and r_i for all $i \neq j$ also have pairwise distinct lambda labels because they already did in t.
   - **Sequential contexts.** The derivation of $\mathtt{C}'\langle s\{x := \vec{u}\}\rangle$ has sequential contexts because it is correct. The derivations of r_i have secuential contexts for all i by hypothesis (their derivation in t' is the same that it was in t). Also, Γ is sequential because t is correct.
   - **Sequential types.** Because it is correct, all types in the derivation of $\mathtt{C}'\langle s\{x := \vec{u}\}\rangle$ are sequential. The derivations of r_i have secuential types for all i by hypothesis (same as before). Finally, if τ is an arrow type, then the domain is sequential because t is correct.

4. **Right of an application,** $\mathtt{C} = r\,[r_1, \ldots, r_{k-1}, \mathtt{C}', r_{k+1}, \ldots, r_n]$**.**
   So $t = r\,[r_1, \ldots, r_{k-1}, \mathtt{C}'\langle(\lambda^{\ell} x.s)\vec{u}\rangle, r_{k+1}, \ldots, r_n]$; $t' = r\,[r_1, \ldots, r_{k-1}, \mathtt{C}'\langle s\{x := \vec{u}\}\rangle, r_{k+1}, \ldots, r_n]$.
   By Lemma A.1, $\mathtt{C}'\langle(\lambda^{\ell} x.s)\vec{u}\rangle$ is correct, so by i.h. $\mathtt{C}'\langle s\{x := \vec{u}\}\rangle$ is correct.

Let $\Gamma \vdash \mathtt{C}'\langle(\lambda^\ell x.s)\vec{u}\rangle\,\vec{r} : \tau$.
By Subject reduction,

$$\frac{\Gamma_0 \vdash r : [\sigma_1,\dots,\sigma_n] \xrightarrow{\ell} \tau \quad (\Gamma_i \vdash r_i : \sigma_i)_{i=1}^{k-1} \quad \Gamma_k \vdash \mathtt{C}'\langle s\{x := \vec{u}\}\rangle : \sigma_k \quad (\Gamma_i \vdash r_i : \sigma_i)_{i=k+1}^{n}}{\Gamma = \Gamma_0 +_{i=1}^{n} \Gamma_i \vdash r\,[r_1,\dots,r_{k-1},\mathtt{C}'\langle s\{x := \vec{u}\}\rangle, r_{k+1},\dots,r_n] : \tau}\ \to_{\mathrm{E}}$$

Then $r\,[r_1,\dots,r_{k-1},\mathtt{C}'\langle s\{x := \vec{u}\}\rangle, r_{k+1},\dots,r_n]$ is correct, because

- **Unique lambdas.** As it is correct, $\mathtt{C}'\langle s\{x := \vec{u}\}\rangle$ enjoys uniqueness of lambda labeling. Also, as t as correct, the labels that decorate the lambdas of r and of all the r_i for $i \neq j$ do not decorate any lambda in $\mathtt{C}'\langle(\lambda^\ell x.s)\vec{u}\rangle$ so by Lemma A.2, they do not decorate any lambda in $\mathtt{C}'\langle s\{x := \vec{u}\}\rangle$. r and r_i for all $i \neq j$ also have pairwise distinct lambda labels because they already did in t.
- **Sequential contexts.** The derivation of $\mathtt{C}'\langle s\{x := \vec{u}\}\rangle$ has sequential contexts because it is correct. The derivations of r and all r_i with $i \neq j$ have sequential contexts by hypothesis (their derivation in t' is the same that it was in t). Also, Γ is sequential because t is correct.
- **Sequential types.** Because it is correct, all types in the derivation of $\mathtt{C}'\langle s\{x := \vec{u}\}\rangle$ are sequential. The derivations of r and all r_i with $i \neq j$ have sequential types by hypothesis (same as before). Finally, if τ is an arrow type, then the domain is sequential because t is correct.

□

A.4 Proof of Lemma 2.20 — Subject reduction

We are to prove that if $\mathtt{C}\langle(\lambda^\ell x.t)\vec{s}\rangle$ is correct then $\mathtt{C}\langle t\{x := \vec{s}\}\rangle$ is correct, and, moreover, that their unique typings are under the same typing context and have the same type. Also, we check that correctness is preserved. We need a few auxiliary results:

Lemma A.3. *If $\vec{s}$ is a permutation of $\vec{u}$, then $t\{x := \vec{s}\} = t\{x := \vec{u}\}$.*

Proof. By induction on t. □

Lemma A.4. *If t is correct, then any subterm of t is correct.*

Proof. Note, by definition of correctness (Definition 2.5), that if t is a correct abstraction $t = \lambda^\ell x.s$ then s is correct, and if t is a correct application $t = s[u_1,\dots,u_n]$ then s and $u_1,\dots,u_n$ are correct. This allows us to conclude by induction on t. □

Lemma A.5 (Relevance). *If $\Gamma \vdash t : \tau$ and $x \in \operatorname{dom}\Gamma$ then $x \in \mathsf{fv}(t)$.*

Proof. By induction on t. □

Definition A.6. $\Lambda(t)$ stands for the multiset of labels decorating the lambdas of t:

$$\begin{array}{rcl} \Lambda(x^\tau) & \stackrel{\text{def}}{=} & [\,] \\ \Lambda(\lambda^\ell x.t) & \stackrel{\text{def}}{=} & [\ell] + \Lambda(t) \\ \Lambda(t[s_i]_{i=1}^n) & \stackrel{\text{def}}{=} & \Lambda(t) +_{i=1}^n \Lambda(s_i) \end{array}$$

For example, $\Lambda((\lambda^1 x.x^{[\alpha^2]\xrightarrow{3}\alpha^2})[\lambda^3 x.x^{\alpha^2}]) = [1,3]$.

Lemma A.7. *Let $t, s_1, \ldots, s_n$ be correct terms. Then $\Lambda(t\{x := [s_i]_{i=1}^n\}) = \Lambda(t) +_{i=1}^n \Lambda(s_i)$.*

Proof. By induction on t. □

To prove Lemma 2.20, we first show that substitution preserves typing, and then that it preserves correctness.

Substitution preserves typing

More precisely, let us show that if $\Gamma \vdash \mathtt{C}\langle(\lambda^\ell x.t)\vec{s}\rangle : \tau$ is derivable, then $\Gamma \vdash \mathtt{C}\langle t\{x := \vec{s}\}\rangle : \tau$ is derivable. By induction on the context $\mathtt{C}$.

1. **Empty, $\mathtt{C} = \Box$.** By induction on t.
   - 1.1 **Variable (same), $t = x^\tau$, $\vec{s} = [s]$.** We have that $x : [\tau] \vdash x^\tau : \tau$ and $\Delta \vdash s : \sigma$ are derivable, so we are done.
   - 1.2 **Variable (different), $t = y^\tau$, $y \neq x$, $\vec{s} = []$.** We have that $y : [\tau] \vdash y^\tau : \tau$ is derivable, so we are done.
   - 1.3 **Abstraction, $t = \lambda^\ell y.u$.** Let $\Gamma \oplus x : [\sigma_i]_{i=1}^n \vdash \lambda^\ell x.u : \mathcal{M} \xrightarrow{\ell} \tau$ be derivable and $\Delta_i \vdash s_i : \sigma_i$ be derivable for all $i = 1..n$. By inversion of the $\to_{\mathrm{I}}$ rule, we have that $\Gamma \oplus y : \mathcal{M} \oplus x : [\sigma_i]_{i=1}^n \vdash u : \tau$ is derivable, so by *i.h.* $(\Gamma \oplus y : \mathcal{M}) +_{i=1}^n \Delta_i \vdash u\{x := [s_i]_{i=1}^n\} : \tau$ is derivable. Observe that $y \notin \mathrm{fv}(s_i)$ so $y \notin \mathrm{dom}\,\Delta_i$ by Lemma A.5. Hence the previous judgment can be written as $(\Gamma +_{i=1}^n \Delta_i) \oplus y : \mathcal{M} \vdash u\{x := [s_i]_{i=1}^n\} : \tau$. Applying the $\to_{\mathrm{I}}$ rule we obtain $\Gamma +_{i=1}^n \Delta_i \vdash \lambda^\ell y.u\{x := [s_i]_{i=1}^n\} : \mathcal{M} \xrightarrow{\ell} \tau$ as required.
   - 1.4 **Application, $t = u[r_j]_{j=1}^m$.** Let $\Gamma \oplus x : [\sigma_i]_{i=1}^n \vdash u[r_j]_{j=1}^m : \tau$ be derivable and $\Delta_i \vdash s_i : \sigma_i$ be derivable for all $i = 1..n$. By inversion of the $\to_{\mathrm{E}}$ rule, the multiset of types $[\sigma_i]_{i=1}^n$ may be partitioned as $[\sigma_i]_{i=1}^n = \sum_{j=0}^m \mathcal{M}_j$, and the typing context Γ may be partitioned as $\Gamma = \sum_{j=0}^m \Gamma_j$ in such a way that $\Gamma_0 \oplus x : \mathcal{M}_0 \vdash u : [\rho_j]_{j=1}^m \xrightarrow{\ell} \tau$ is derivable and $\Gamma_j \oplus x : \mathcal{M}_j \vdash r_j : \rho_j$ is derivable for all $j = 1..m$. Consider a partition $(\vec{s}_0, \ldots, \vec{s}_j)$ of the list $\vec{s}$ such that for every $j = 0..m$ we have $\mathtt{T}(\vec{s}_j) = \mathcal{M}_j$. Observe that this partition exists since $\mathtt{T}(\vec{s}_0 + \ldots + \vec{s}_j) = \mathtt{T}(\vec{s}) = \sum_{j=0}^m \mathcal{M}_j = [\sigma_i]_{i=1}^n = \mathtt{T}_x(t)$.
     Moreover, let $\Theta_j = \sum_{i:s_i \in \vec{s}_j} \Delta_i$ for all $j = 0..m$. By *i.h.* we have that $\Gamma_0 + \Theta_0 \vdash u\{x := \vec{s}_0\} : [\rho_j]_{j=1}^m \xrightarrow{\ell} \tau$ is derivable and $\Gamma_j + \Theta_j \vdash r_j\{x := \vec{s}_j\} : \rho_j$ is derivable for all $j = 1..m$. Applying the $\to_{\mathrm{E}}$ rule we obtain that $\sum_{j=0}^m \Gamma_j + \sum_{j=0}^m \Theta_j \vdash u[r_j]\{x := \sum_{j=0}^m \vec{s}_j\} : \tau$ is derivable. By definition of $\Gamma_0, \ldots, \Gamma_m$ and $\Theta_0, \ldots, \Theta_m$ this judgment equals $\Gamma +_{i=1}^n \Delta_i \vdash u[r_j]\{x := \sum_{j=0}^m \vec{s}_j\} : \tau$. By definition of $\vec{s}_0, \ldots, \vec{s}_m$ and Lemma A.3 this in turn equals $\Gamma +_{i=1}^n \Delta_i \vdash u[r_j]\{x := \vec{s}\} : \tau$, as required.
2. **Under an abstraction, $\mathtt{C} = \lambda^{\ell'} y.\mathtt{C}'$.** Straightforward by *i.h.*.
3. **Left of an application, $\mathtt{C} = \mathtt{C}'\vec{u}$.** Straightforward by *i.h.*.
4. **Right of an application, $\mathtt{C} = u[\vec{r}_1, \mathtt{C}', \vec{r}_2]$.** Straightforward by *i.h.*.

Substitution preserves correctness

More precisely, let us show that if $\mathtt{C}\langle(\lambda^\ell x.t)\vec{s}\rangle$ is correct then $\mathtt{C}\langle t\{x := \vec{s}\}\rangle$ is correct. This is true by Lemma 2.20.

A.5 Proof of Proposition 2.21 (cont.) — Termination

We wanted to prove that $d(t) + \sum_{i=1}^{n} d(s_i) = d(t\{x := \vec{s}\})$, where d counts the number of lambdas of its paramter.
Proof. We proceed by induction on t.

- **Variable (same).** $t = x^\tau$. Then, $\vec{s} = [s]$ and $x^\tau\{x := [s]\} = s$. So what we want to show is true:

$$d(x^\tau) + d(s) = d(s)$$

- **Variable (different).** $t = y^\tau \neq x$. Then, $\vec{s} = [\,]$ and $y^\tau\{x := [\,]\} = y^\tau$. So:

$$d(y^\tau) + 0 = d(y^\tau)$$

- **Abstraction.** $t = \lambda^\ell y.r$. Then, $(\lambda^\ell y.r)\{x := \vec{s}\} = \lambda^\ell y.r\{x := \vec{s}\}$.

  By inductive hypothesis on r and $\vec{s}$, we have that $d(r) + \sum_{i=1}^{n} d(s_i) = d(r\{x := \vec{s}\})$, but that is what we needed, because

$$d(\lambda^\ell y.r) + \sum_{i=1}^{n} d(s_i) = 1 + d(r) + \sum_{i=1}^{n} d(s_i) = 1 + d(r\{x := \vec{s}\}) = d(\lambda^\ell y.r\{x := \vec{s}\})$$

- **Application.** $t = u[r_i]_{i=1}^{n}$. Then,

$$u[r_i]_{i=1}^{n}\{x := \vec{s}\} = u\{x := \vec{s}_0\}[r_i\{x := \vec{s}_i\}]_{i=1}^{n}$$

  where $\vec{s}_0 +_{i=1}^{n} \vec{s}_i$ is a permutation of $\vec{s}$ such that $\mathtt{T}_x(u) = \mathtt{T}(\vec{s}_0)$ and $\mathtt{T}_x(r_i) = \mathtt{T}(\vec{s}_i)$ for all $i \in \{1, \ldots, n\}$. Let $l(i) = |\vec{s}_i|$.

  By inductive hypothesis

  - on u and $\vec{s}_0$, we have that $d(u) + \sum_{j=1}^{l(0)} d(\vec{s}_{0,j}) = d(u\{x := \vec{s}_0\})$
  - on r_i and $\vec{s}_i$ (for all $i \in \{1, \ldots, n\}$), we have that $d(r_i) + \sum_{j=1}^{l(i)} d(\vec{s}_{i,j}) = d(r_i\{x := \vec{s}_i\})$

  Note that $\vec{s} = \sum_{i=0}^{n} \sum_{j=1}^{l(i)} \vec{s}_{i,j}$.

Using all that we can finally show what we wanted:

$$\begin{aligned}
d(u[r_i]_{i=1}^n) + d(\vec{s}) &= d(u) + \sum_{i=1}^{n} d(r_i) + \sum_{i=0}^{n}\sum_{j=1}^{l(i)} d(\vec{s}_{i,j}) \\
&= d(u) + \sum_{j=1}^{l(0)} d(\vec{s}_{0,j}) + \sum_{i=1}^{n} d(r_i) + \sum_{i=1}^{n}\sum_{j=1}^{l(i)} d(\vec{s}_{i,j}) \\
&= d(u) + \sum_{j=1}^{l(0)} d(\vec{s}_{0,j}) + \sum_{i=1}^{n} \left(d(r_i) + \sum_{j=1}^{l(i)} d(\vec{s}_{i,j}) \right) \\
&= d(u\{x := \vec{s}_0\}) + \sum_{i=1}^{n} d(r_i\{x := \vec{s}_i\}) \\
&= d(u\{x := \vec{s}_0\}[r_i\{x := \vec{s}_i\}]_{i=1}^n) \\
&= d(u[r_i]_{i=1}^n\{x := \vec{s}\})
\end{aligned}$$

□

A.6 Proof of Lemma 2.25 — Substitution lemma

We want to prove that if $x \neq y$ and $x \notin \mathrm{fv}(\vec{u})$, then

$$t\{x := \vec{s}\}\{y := \vec{u}\} = t\{y := \vec{u}_1\}\{x := \vec{s}\{y := \vec{u}_2\}\}$$

where $\vec{u}$ is a permutation of $\vec{u}_1 + \vec{u}_2$.

Recall that the equation above requires that both sides be defined; in particular $|\vec{u}_1| = \#_y(t)$ and $|\vec{u}_2| = \#_y(\vec{s})$. When read from left to right, the equation means that given $\vec{u}$, it can always be written as some permutation of $\vec{u}_1 + \vec{u}_2$. When read from right to left, it means that given $\vec{u}_1$ and $\vec{u}_2$, taking $\vec{u}$ to be any permutation of $\vec{u}_1 + \vec{u}_2$ verifies the equality.

Proof. By induction on t.

1. **Variable (same, first case)** $t = x$**.** Then $\vec{s} = [s]$ and $\vec{u}_1 = [\,]$, so $\vec{u}_2 = \vec{u}$.
   Firstly,

   $$x\{x := \vec{s}\}\{y := \vec{u}\} = s\{y := \vec{u}\}$$

   Also,

   $$\begin{aligned}
   x\{y := []\}\{x := [s]\{y := \vec{u}\}\} &= x\{x := [s]\{y := \vec{u}\}\} \\
   &= x\{x := [s\{y := \vec{u}\}]\} \\
   &= s\{y := \vec{u}\}
   \end{aligned}$$

2. **Variable (same, second case)** $t = y$**.** Then $\vec{s} = [\,]$, so $\vec{u} = [u]$.
   Firstly,

   $$\begin{aligned}
   y\{x := [\,]\}\{y := [u]\} &= y\{y := [u]\} \\
   &= u
   \end{aligned}$$

Also,

$$\begin{aligned} y\{y := [u]\}\{x := [\,]\{y := [\,]\}\} &= u\{x := [\,]\{y := [\,]\}\} \\ &= u\{x := [\,]\} \\ &= u \end{aligned}$$

3. **Variable (different)** $t = z$**.** Then $\vec{s} = [\,]$ and $\vec{u} = [\,]$.
Firstly,

$$\begin{aligned} z\{x := [\,]\}\{y := [\,]\} &= z\{y := [\,]\} \\ &= z \end{aligned}$$

Also,

$$\begin{aligned} z\{y := [\,]\}\{x := [\,]\{y := [\,]\}\} &= z\{x := [\,]\{y := [\,]\}\} \\ &= z\{x := [\,]\} \\ &= z \end{aligned}$$

4. **Abstraction,** $t = \lambda^{\ell} z.r$**.**

$$\begin{aligned} (\lambda^{\ell} z.r)\{x := \vec{s}\}\{y := \vec{u}\} &= (\lambda^{\ell} z.r\{x := \vec{s}\})\{y := \vec{u}\} \\ &= \lambda^{\ell} z.r\{x := \vec{s}\}\{y := \vec{u}\} \\ &= \lambda^{\ell} z.r\{x := \vec{s}\}\{y := \vec{u}\} \\ &\overset{\text{h.i.}}{=} \lambda^{\ell} z.r\{y := \vec{u}_1\}\{x := \vec{s}\{y := \vec{u}_2\}\} \\ &= (\lambda^{\ell} z.r\{y := \vec{u}_1\})\{x := \vec{s}\{y := \vec{u}_2\}\} \\ &= (\lambda^{\ell} z.r)\{y := \vec{u}_1\}\{x := \vec{s}\{y := \vec{u}_2\}\} \end{aligned}$$

5. **Application,** $t = r[r_1, \ldots, r_n]$**.**

$$\begin{aligned} (r[r_1, \ldots, r_n])\{x := \vec{s}\}\{y := \vec{u}\} &= (r\{x := \vec{s}_0\}[r_i\{x := \vec{s}_i\}]_{i=0}^{n})\{y := \vec{u}\} \\ &= r\{x := \vec{s}_0\}\{y := \vec{u}_0\}[r_i\{x := \vec{s}_i\}\{y := \vec{u}_i\}]_{i=0}^{n} \\ &\overset{\text{h.i.}}{=} r\{y := \vec{u}_{0,1}\}\{x := \vec{s}_0\{y := \vec{u}_{0,2}\}\}[r_i\{y := \vec{u}_{i,1}\}\{x := \vec{s}_0\{y := \vec{u}_{i,2}\}\}]_{i=0}^{n} \\ &= (r\{y := \vec{u}_{0,1}\}[r_i\{y := \vec{u}_{i,1}\}]_{i=0}^{n})\{x := \sum_{i=1}^{n} \vec{s}_i\{y := \vec{u}_{i,2}\}\} \end{aligned}$$

Because $\sum_{i=0}^{n} \vec{s}_i$ is a permutation of $\vec{s}$ and we can perform the substitution of each $\vec{u}_{i,2}$ simultaneously, defining $\vec{v}_2$ as $\sum_{i=0}^{n} \vec{u}_{i,2}$, the last term can be rewritten to:

$$(r\{y := \vec{u}_{0,1}\}[r_i\{y := \vec{u}_{i,1}\}]_{i=0}^{n})\{x := \vec{s}\{y := \vec{v}_2\}\}$$

Lastly, if we define $\vec{v}_1$ to be $\sum_{i=0}^{n} \vec{u}_{i,1}$, we get the desired term:

$$(r[r_1, \ldots, r_n])\{y := \vec{v}_1\}\{x := \vec{s}\{y := \vec{v}_2\}\}$$

To conclude observe that $\vec{v}_1 + \vec{v}_2$ is indeed a permutation of $\vec{u}$; indeed:

$$\begin{array}{lcll} \vec{v}_1 + \vec{v}_2 & = & (\sum_{i=0}^{n} \vec{u}_{i,1}) + (\sum_{i=0}^{n} \vec{u}_{i,2}) & \\ & \approx & \sum_{i=0}^{n} \vec{u}_{i,1} + \vec{u}_{i,2} & \\ & \approx & \sum_{i=0}^{n} \vec{u}_i & \text{by } \textit{i.h.} \text{ on each index } i = 0..n \\ & = & \vec{u} & \end{array}$$

□

A.7 Proof of Proposition 2.27 — Strong Permutation

Recall that we had two steps $R : t_0 \xrightarrow{\ell}_{\#} t_1$ and $S : t_0 \xrightarrow{\ell'}_{\#} t_2$, going out from t_0, and we wanted to show that t_1 and t_2 have one reduct in common one step away.

We use some auxiliary lemmas that are stated and proven at the end of this section.

Proof. We suppose the step R is of the form:

$$R : t_0 = \mathtt{C}\langle(\lambda^{\ell} x.t)\vec{s}\rangle \xrightarrow{\ell}_{\#} \mathtt{C}\langle t\{x := \vec{s}\}\rangle = t_1$$

Recall that $R \neq S$ by hypothesis, so we know it must reduce some other lambda. We proceed by induction on $\mathtt{C}$.

1. **Empty context,** $\mathtt{C} = \Box$**.** Then

$$R : t_0 = (\lambda^{\ell} x.t)\vec{s} \xrightarrow{\ell}_{\#} t\{x := \vec{s}\} = t_1$$

There are two subcases, depending on whether the redex S is inside t or inside $\vec{s}$:

1.1 S **is inside** t**.**

$$\begin{array}{ccc}
(\lambda^{\ell} x.\mathtt{C}\langle(\lambda^{\ell'} y.r)\vec{u}\rangle)\vec{s} & \xrightarrow{\quad\ell'\quad} & (\lambda^{\ell} x.\mathtt{C}\langle r\{y := \vec{u}\}\rangle)\vec{s} \\
\downarrow \ell & & \downarrow \ell \\
\mathtt{C}\langle(\lambda^{\ell'} y.r)\vec{u}\rangle\{x := \vec{s}\} & & \mathtt{C}\langle r\{y := \vec{u}\}\rangle\{x := \vec{s}\} \\
\| & & \| \\
\mathtt{C}\{x := \vec{s}_1\}\langle(\lambda^{\ell'} y.r\{x := \vec{s}_2\})\vec{u}\{x := \vec{s}_3\}\rangle & & \mathtt{C}\{x := \vec{s}_1\}\langle r\{y := \vec{u}\}\{x := \vec{s}_0\}\rangle \\
\downarrow \ell' & & \\
\mathtt{C}\{x := \vec{s}_1\}\langle r\{x := \vec{s}_2\}\{y := \vec{u}\{x := \vec{s}_3\}\}\rangle & &
\end{array}$$

Note that $\vec{s}_2 + \vec{s}_3$ is a permutation of $\vec{s}_0$. It is enough to see that

$$r\{y := \vec{u}\}\{x := \vec{s}_0\} = r\{x := \vec{s}_2\}\{y := \vec{u}\{x := \vec{s}_3\}\}$$

But that is true because of the Substitution lemma (Lemma 2.25).

1.2 S **is inside** $\vec{s}$**.** In this case,

$$\vec{s} = [s_1, \ldots, s_{j-1}, \mathtt{C}\langle(\lambda^{\ell'} y.r)\vec{u}\rangle, s_{j+1}, \ldots, s_n] = [\vec{s}_{1:j-1}, \mathtt{C}\langle(\lambda^{\ell'} y.r)\vec{u}\rangle, \vec{s}_{j+1:n}]$$

$$\begin{array}{ccc}
(\lambda^{\ell} x.t)[\vec{s}_{1:j-1}, \mathtt{C}\langle(\lambda^{\ell'} y.r)\vec{u}\rangle, \vec{s}_{j+1:n}] & \xrightarrow{\quad\ell'\quad} & (\lambda^{\ell} x.t)[\vec{s}_{1:j-1}, r\{y := \vec{u}\}, \vec{s}_{j+1:n}] \\
\downarrow \ell & & \downarrow \ell \\
t\{x := [\vec{s}_{1:j-1}, \mathtt{C}\langle(\lambda^{\ell'} y.r)\vec{u}\rangle, \vec{s}_{j+1:n}]\} & & t\{x := [\vec{s}_{1:j-1}, r\{y := \vec{u}\}, \vec{s}_{j+1:n}]\}
\end{array}$$

We can close the diagram reducing from left to right using Lemma A.8.

2. **Under an abstraction,** $\mathtt{C} = \lambda^{\ell''} y.\mathtt{C}'$**.** Then for some t_1 and t_2,

$$\begin{array}{ccc} \lambda^{\ell''} y.\mathtt{C}'\langle(\lambda^{\ell} x.t)\vec{s}\rangle & \xrightarrow{\ell'} & \lambda^{\ell''} y.t_2 \\ \downarrow{\scriptstyle \ell} & & \\ \lambda^{\ell''} y.t_1 & & \end{array}$$

But by inductive hypothesis,

$$\begin{array}{ccc} \mathtt{C}'\langle(\lambda^{\ell} x.t)\vec{s}\rangle & \xrightarrow{\ell'} & t_2 \\ \downarrow{\scriptstyle \ell} & & \vdots{\scriptstyle \ell} \\ t_1 & \overset{\ell'}{\dashrightarrow} & t_3 \end{array}$$

So if we consider those steps in the full term we get our desired diagram.

$$\begin{array}{ccc} \lambda^{\ell''} y.\mathtt{C}'\langle(\lambda^{\ell} x.t)\vec{s}\rangle & \xrightarrow{\ell'} & \lambda^{\ell''} y.t_2 \\ \downarrow{\scriptstyle \ell} & & \vdots{\scriptstyle \ell} \\ \lambda^{\ell''} y.t_1 & \overset{\ell'}{\dashrightarrow} & \lambda^{\ell''} y.t_3 \end{array}$$

3. **Left of an application,** $\mathtt{C} = \mathtt{C}'\vec{u}$**.** There are three subcases, depending on whether the redex S is at the root, to the left of the application, or to the right of the application.

   3.1 S **is inside** $\mathtt{C}'$**.** Then by inductive hypothesis, we have the following diagram for some t_1 and t_2:

$$\begin{array}{ccc} \mathtt{C}'\langle(\lambda^{\ell} x.t)\vec{s}\rangle & \xrightarrow{\ell'} & t_2 \\ \downarrow{\scriptstyle \ell} & & \vdots{\scriptstyle \ell} \\ t_1 & \overset{\ell'}{\dashrightarrow} & t_3 \end{array}$$

   So if we consider those steps in the full term we get our desired diagram.

$$\begin{array}{ccc} \mathtt{C}'\langle(\lambda^{\ell} x.t)\vec{s}\rangle\vec{u} & \xrightarrow{\ell'} & t_2\vec{u} \\ \downarrow{\scriptstyle \ell} & & \vdots{\scriptstyle \ell} \\ t_1\vec{u} & \overset{\ell'}{\dashrightarrow} & t_3\vec{u} \end{array}$$

   3.2 S **is inside** $\vec{u}$**.** What we have can be expressed in the following diagram.

$$\begin{array}{ccc} \mathtt{C}'\langle(\lambda^{\ell} x.t)\vec{s}\rangle[\vec{s}_{1:j-1}, \mathtt{C}''\langle(\lambda^{\ell'} y.r)\vec{u}\rangle, \vec{s}_{j+1:n}] & \xrightarrow{\ell'} & (\lambda^{\ell} x.t)[\vec{s}_{1:j-1}, r\{y := \vec{u}\}, \vec{s}_{j+1:n}] \\ \downarrow{\scriptstyle \ell} & & \downarrow{\scriptstyle \ell} \\ t\{x := [\vec{s}_{1:j-1}, \mathtt{C}''\langle(\lambda^{\ell'} y.r)\vec{u}\rangle, \vec{s}_{j+1:n}]\} & & t\{x := [\vec{s}_{1:j-1}, r\{y := \vec{u}\}, \vec{s}_{j+1:n}]\} \end{array}$$

   We can close the diagram reducing from left to right using Lemma A.8.

3.3 S **is at the root**. What this means is that $\mathtt{C}'\langle(\lambda^{\ell'}x.t)\vec{s}\rangle$ has a lambda at its root, so it is equal to $\lambda^{\ell'}y.\mathtt{C}'''\langle(\lambda^{\ell}x.t)\vec{s}\rangle$ for some, $\mathtt{C}'''$ and y.
Hence, the term we have is $(\lambda^{\ell'}y.\mathtt{C}'''\langle(\lambda^{\ell}x.t)\vec{s}\rangle)\vec{u}$, which has already been dealt with in case 1.1.

4. **Right of an application,** $\mathtt{C} = r[u_1,\ldots,u_{i-1},\mathtt{C}',u_{i+1},\ldots,u_n] = r[u_{1:i-1},\mathtt{C}',u_{i+1:n}]$. There are four subcases, depending on whether the redex S is at the root, to the left of the application (that is, inside r), or to the right of the application (that is, either inside $\mathtt{C}'$ or u_j for some j).
   4.1 S **is inside** r**.** Notice that the steps are disjoint, so we may close the diagram in a straightforward manner.
   4.2 S **is inside** u_j **for some** $j \neq i$**.** Notice that, again, the steps are disjoint, so we may close the diagram in a straightforward manner.
   4.3 S **is inside** $\mathtt{C}'$**.** Then by inductive hypothesis, we have the following diagram for some t_1 and t_2:

$$\begin{array}{ccc} \mathtt{C}'\langle(\lambda^{\ell}x.t)\vec{s}\rangle & \xrightarrow{\ell'} & t_2 \\ \downarrow{\scriptstyle \ell} & & \downarrow{\scriptstyle \ell} \\ t_1 & \overset{\ell'}{\dashrightarrow} & t_3 \end{array}$$

   So if we consider those steps in the full term we get our desired diagram.

$$\begin{array}{ccc} r[u_{1:i-1},\mathtt{C}'\langle(\lambda^{\ell}x.t)\vec{s}\rangle,u_{i+1:n}] & \xrightarrow{\ell'} & r[u_{1:i-1},t_2,u_{i+1:n}] \\ \downarrow{\scriptstyle \ell} & & \downarrow{\scriptstyle \ell} \\ r[u_{1:i-1},t_1,u_{i+1:n}] & \overset{\ell'}{\dashrightarrow} & r[u_{1:i-1},t_3,u_{i+1:n}] \end{array}$$

   4.4 S **is at the root.** Hence t has a lambda at the root, so $r[u_{1:i-1},\mathtt{C}'\langle(\lambda^{\ell}x.t)s\rangle,u_{i+1:n}] = (\lambda^{\ell'}y.u)[u_{1:i-1},\mathtt{C}'\langle(\lambda^{\ell}x.t)s\rangle,u_{i+1:n}]$, which is the same case we already proved in 1.2. □

Auxiliary lemma

Lemma A.8 (Reduction)**.** *Suppose that* $s_k \xrightarrow{\ell}_{\#} s'_k$*. Then if* $t\{x := [s_1,\ldots,s_{k-1},s_k,s_{k+1},\ldots,s_n]\}$ *is defined, so is* $t\{x := [s_1,\ldots,s_{k-1},s'_k,s_{k+1},\ldots,s_n]\}$ *and*

$$t\{x := [s_1,\ldots,s_{k-1},s_k,s_{k+1},\ldots,s_n]\} \xrightarrow{\ell}_{\#} t\{x := [s_1,\ldots,s_{k-1},s'_k,s_{k+1},\ldots,s_n]\}$$

Proof. By induction on t. We will write $[s_1,\ldots,s_{k-1},s_k,s_{k+1},\ldots,s_n]$ as $[s_{1:k-1},s_k,s_{k+1:n}]$

1. **Variable (same)** $t = x$**.**

$$\begin{aligned} x\{x := [s_k]\} &= s_k \\ &\xrightarrow{\ell}_{\#} s'_k \\ &= x\{x := [s'_k]\} \end{aligned}$$

2. **Variable (different)** $t = y$**.** This case cannot happen because $[s_1,\ldots,s_n]$ must be empty but at the same time must contain s_k.

3. **Abstraction,** $t = \lambda^{\ell} y.r$**.**

$$\begin{aligned}(\lambda^{\ell} y.r)\{x := [s_{1:k-1}, s_k, s_{k+1:n}]\} &= \lambda^{\ell} y.r\{x := [s_{1:k-1}, s_k, s_{k+1:n}]\} \\ &\overset{\text{(h.i.)}}{\to_{\#}} \lambda^{\ell} y.r\{x := [s_{1:k-1}, s'_k, s_{k+1:n}]\} \\ &= (\lambda^{\ell} y.r)\{x := [s_{1:k-1}, s'_k, s_{k+1:n}]\}\end{aligned}$$

4. **Application,** $t = r[r_1, \ldots, r_m]$**.**

$$(r[r_1, \ldots, r_m])\{x := [s_{1:k-1}, s_k, s_{k+1:n}]\} = r\{x := \vec{s}_0\}[r_i\{x := \vec{s}_i\}]_{i=0}^{m}$$

Where s_k is one of the terms in the list $\vec{s}_i$ for some $i \in \{0, \ldots, m\}$. Suppose that it is in $\vec{s}_0$ (if it is in $\vec{s}_i$ for some $i > 0$, it is analogous). This means that $\vec{s}_0 = [s_{0,1}, \ldots, s_k, \ldots, s_{0,l}]$, where $l = |\vec{s}_0|$.
By inductive hypothesis,

$$\begin{aligned}r\{x := [s_{0,1}, \ldots, s_k, \ldots, s_{0,l}]\}[r_i\{x := \vec{s}_i\}]_{i=0}^{m} &\xrightarrow{\ell}_{\#} r\{x := [s_{0,1}, \ldots, s'_k, \ldots, s_{0,l}]\}[r_i\{x := \vec{s}_i\}]_{i=0}^{m} \\ &= (r[r_1, \ldots, r_m])\{x := [s_{0,1}, \ldots, s'_k, \ldots, s_{0,l}] + \vec{s}_1 + \ldots + \vec{s}_m\} \\ &= (r[r_1, \ldots, r_m])\{x := [s_{1:k-1}, \ldots, s'_k, \ldots, s_{k+1:n}]\}\end{aligned}$$

□

A.8 Proof of Lemma 3.18 — Creation

Proof. We wanted to check that the cases of creation in the distributive lambda-calculus where

1. **Creation case I.** $\mathtt{C}\langle(\lambda^{\ell} x.x^{\tau})\,[\lambda^{\ell'} y.t]\,\vec{s}\rangle \to_{\#} \mathtt{C}\langle(\lambda^{\ell'} y.t)\,\vec{s}\rangle \to_{\#} \mathtt{C}\langle t\{\!\{y := \vec{s}\}\!\}\rangle$.
2. **Creation case II.** $\mathtt{C}\langle(\lambda^{\ell} x.\lambda^{\ell'} y.t)\,\vec{s}\,\vec{u}\rangle \to_{\#} \mathtt{C}\langle(\lambda^{\ell'} y.t')\,\vec{u}\rangle \to_{\#} \mathtt{C}\langle t'\{\!\{y := \vec{u}\}\!\}\rangle$, where:

$$t' \;=\; t\{\!\{x := \vec{s}\}\!\}$$

3. **Creation case III.** $\mathtt{C}_1\langle(\lambda^{\ell} x.\mathtt{C}_2\langle x^{\tau}\,\vec{t}\rangle)\vec{s}\rangle \to_{\#} \mathtt{C}_1\langle\mathtt{C}'_2\langle(\lambda^{\ell'} y.u)\,\vec{t'}\rangle\rangle \to_{\#} \mathtt{C}_1\langle\mathtt{C}'_2\langle u\{\!\{y := \vec{t'}\}\!\}\rangle\rangle$, where:

$$\begin{aligned}\mathtt{C}_2\{\!\{x := \vec{s}\}\!\} &= \mathtt{C}'_2 \\ x^{\tau}\{\!\{x := \vec{s}\}\!\} &= \lambda^{\ell'} y.u \\ \vec{t}\{\!\{x := \vec{s}\}\!\} &= \vec{t'} \\ \vec{s} &= [\vec{s_1}, \lambda^{\ell'} y.u, \vec{s_2}]\end{aligned}$$

We supposed that $R : \mathtt{C}\langle(\lambda^{\ell} x.t)\vec{s}\rangle \to_{\#} \mathtt{C}\langle t\{\!\{x := \vec{s}\}\!\}\rangle$ wasstep, and $S : \mathtt{C}\langle t\{\!\{x := \vec{s}\}\!\}\rangle \to_{\#} p$ was another step such that R creates S. The said that the redex contracted by the step S is below a context $\mathtt{C}_1$, and $\mathtt{C}\langle t\{\!\{x := \vec{s}\}\!\}\rangle = \mathtt{C}_1\langle(\lambda^{\ell'} y.u)\vec{r}\rangle$, where $(\lambda^{\ell'} y.u)\vec{r}$ is the redex contracted by S.

We are to consider three cases, depending on the relative positions of the holes of $\mathtt{C}$ and $\mathtt{C}_1$, namely they may be disjoint, $\mathtt{C}$ may be a prefix of $\mathtt{C}_1$, or $\mathtt{C}_1$ may be a prefix of $\mathtt{C}$:

1. **If the holes of** $\mathtt{C}$ **and** $\mathtt{C}_1$ **are disjoint.** Then there is a two-hole context $\hat{\mathtt{C}}$ such that

$$\mathtt{C} = \hat{\mathtt{C}}\langle\Box, (\lambda^{\ell'} y.u)\vec{r}\rangle \qquad \mathtt{C}_1 = \hat{\mathtt{C}}\langle t\{\!\{x := \vec{s}\}\!\}, \Box\rangle$$

So the situation is the following:

$$\hat{\mathtt{C}}\langle(\lambda^{\ell} x.t)\vec{s}, (\lambda^{\ell'} y.u)\vec{r}\rangle \to_{\#} \hat{\mathtt{C}}\langle t\{\!\{x := \vec{s}\}\!\}, (\lambda^{\ell'} y.u)\vec{r}\rangle \to_{\#} \hat{\mathtt{C}}\langle t\{\!\{x := \vec{s}\}\!\}, u\{\!\{y := r\}\!\}\rangle$$

Observe that S has an ancestor S_0, contradicting the hypothesis that R creates S. So this case is impossible.

2. **If $\mathtt{C}$ is a prefix of $\mathtt{C}_1$.** Then there exists a context $\mathtt{C}_2$ such that $\mathtt{C}_1 = \mathtt{C}\langle\mathtt{C}_2\rangle$, and we have that $t\{\!\{x := \vec{s}\}\!\} = \mathtt{C}_2\langle(\lambda^{\ell'} y.u)\vec{r}\rangle$. We consider two subcases, depending on whether the position of the hole $\mathtt{C}_2$ lies inside the term t, or it reaches a free occurrence of x in t and "jumps" to one of the arguments in the list $\vec{s}$.
   2.1 **If the position of the hole of $\mathtt{C}_2$ lies in t.** More precisely, there is a context $\mathtt{C}'_2$ and a term t' such that:
$$\begin{array}{rcl} t & = & \mathtt{C}'_2\langle t'\rangle \\ \mathtt{C}_2 & = & \mathtt{C}'_2\{\!\{x := \vec{s}\}\!\} \\ (\lambda^{\ell'} y.u)\vec{r} & = & t'\{\!\{x := \vec{s}\}\!\} \end{array}$$
   We consider three further subcases for the term t': it cannot be an abstraction, so it may be a variable x^τ, or an application. Besides, if it is an application, the head may be a variable x^τ or an abstraction.
      2.1.1 **Variable, *i.e.* $t' = x^\tau$.** Then the list $\vec{s}$ may be split as $\vec{s} = [\vec{s_1}, (\lambda^{\ell'} y.u)\vec{r}, \vec{s_2}]$, and the situation is:
$$\mathtt{C}\langle(\lambda^{\ell} x.\mathtt{C}'_2\langle x^\tau\rangle)[\vec{s_1}, (\lambda^{\ell'} y.u)\vec{r}, \vec{s_2}]\rangle \;\to_{\#}\; \mathtt{C}\langle\mathtt{C}_2\langle(\lambda^{\ell'} y.u)\vec{r}\rangle\rangle \;\to_{\#}\; \mathtt{C}\langle\mathtt{C}_2\langle u\{\!\{y := \vec{r}\}\!\}\rangle\rangle$$
      Observe that S has an ancestor S_0, contradicting the hypothesis that R creates S. So this case is impossible.
      2.1.2 **Application of a variable, *i.e.* $t' = x^\tau\,\vec{r'}$.** Observe that, since $(\lambda^{\ell'} y.u)\vec{r} = t'\{\!\{x := \vec{s}\}\!\}$, we have that $\vec{r} = \vec{r'}\{\!\{x := \vec{s}\}\!\}$, and the list $\vec{s}$ may be split as $\vec{s} = [\vec{s_1}, \lambda^{\ell'} y.u, \vec{s_2}]$. The situation is:
$$\mathtt{C}\langle(\lambda^{\ell} x.\mathtt{C}'_2\langle x^\tau\,\vec{r'}\rangle)[\vec{s_1}, \lambda^{\ell'} y.u, \vec{s_2}]\rangle \;\to_{\#}\; \mathtt{C}\langle\mathtt{C}_2\langle(\lambda^{\ell'} y.u)\,\vec{r}\rangle\rangle \;\to_{\#}\; \mathtt{C}\langle\mathtt{C}_2\langle u\{\!\{y := \vec{r}\}\!\}\rangle\rangle$$
      and we are in the situation of **Creation case III**.
      2.1.3 **Application of an abstraction, *i.e.* $t' = (\lambda^{\ell} y.u')\,\vec{r'}$.** Observe that, since $(\lambda^{\ell'} y.u)\vec{r} = t'\{\!\{x := \vec{s}\}\!\}$, we have that $\vec{u} = \vec{u'}\{\!\{x := \vec{s}\}\!\}$ and $\vec{r} = \vec{r'}\{\!\{x := \vec{s}\}\!\}$. Then the situation is:
$$\mathtt{C}\langle(\lambda^{\ell} x.\mathtt{C}'_2\langle(\lambda^{\ell} y.u')\,\vec{r'}\rangle)\vec{s}\rangle \;\to_{\#}\; \mathtt{C}\langle\mathtt{C}_2\langle(\lambda^{\ell} y.u)\,\vec{r}\rangle\rangle \;\to_{\#}\; \mathtt{C}\langle\mathtt{C}_2\langle u\{\!\{z := \vec{r}\}\!\}\rangle\rangle$$
      Observe that S has an ancestor S_0, contradicting the hypothesis that R creates S. So this case is impossible.
   2.2 **If the position of the hole of $\mathtt{C}_2$ "jumps" through a free occurrence of x.** More precisely, there exist $\mathtt{C}_3$, τ, $\vec{s_1}$, $\vec{s_2}$, and $\mathtt{C}_4$ such that:
$$t = \mathtt{C}_3\langle x^\tau\rangle \qquad \vec{s} = [\vec{s_1}, \mathtt{C}_4\langle(\lambda^{\ell'} y.u)\vec{r}\rangle, \vec{s_2}]$$
   in such a way that $\mathtt{T}(\mathtt{C}_4\langle(\lambda^{\ell'} y.u)\vec{r}\rangle) = \mathtt{T}_x(x^\tau)$, and $\mathtt{C}_2 = \mathtt{C}_3\{\!\{x := \vec{s}\}\!\}\langle\mathtt{C}_4\rangle$. Hence the situation is:
$$\begin{array}{ll} & \mathtt{C}\langle(\lambda^{\ell} x.\mathtt{C}_3\langle x^\tau\rangle)[\vec{s_1}, \mathtt{C}_4\langle(\lambda^{\ell'} y.u)\vec{r}\rangle, \vec{s_2}]\rangle \\ \to_{\#} & \mathtt{C}\langle\mathtt{C}_3\{\!\{x := \vec{s}\}\!\}\langle\mathtt{C}_4\langle(\lambda^{\ell'} y.u)\vec{r}\rangle\rangle\rangle \\ \to_{\#} & \mathtt{C}\langle\mathtt{C}_3\{\!\{x := \vec{s}\}\!\}\langle\mathtt{C}_4\langle u\{\!\{y := \vec{r}\}\!\}\rangle\rangle\rangle \end{array}$$
   Observe that S has an ancestor S_0, contradicting the hypothesis that R creates S. So this case is impossible.
3. **If $\mathtt{C}_1$ is a prefix of $\mathtt{C}$.** Then there exists a context $\mathtt{C}_2$ such that $\mathtt{C} = \mathtt{C}_1\langle\mathtt{C}_2\rangle$. This means that $\mathtt{C}_2\langle t\{\!\{x := \vec{s}\}\!\}\rangle = (\lambda^{\ell'} y.u)\vec{r}$. We consider three subcases, depending on whether $\mathtt{C}_2$ is empty, or the hole of $\mathtt{C}_2$ lies to the left or to the right of the application.

3.1 **Empty,** $\mathtt{C}_2 = \Box$**.** Then $\mathtt{C} = \mathtt{C}_1$ so $\mathtt{C}$ is a prefix of $\mathtt{C}_1$. We have already considered this case.

3.2 **Left,** $\mathtt{C}_2 = \mathtt{C}_2' \vec{r}$**.** Then the step R is of the form:

$$R : \mathtt{C}_1\langle \mathtt{C}_2'\langle (\lambda^\ell x.t)\vec{s}\rangle \vec{r}\rangle \to_\# \mathtt{C}_1\langle \mathtt{C}_2'\langle t\{\!\{x := \vec{s}\}\!\}\rangle \vec{r}\rangle$$

in particular we know that $\mathtt{C}_2'\langle t\{\!\{x := \vec{s}\}\!\}\rangle = \lambda^{\ell'} y.u$. There are two cases, depending on whether $\mathtt{C}_2'$ is empty or non-empty.

3.2.1 **Empty,** $\mathtt{C}_2' = \Box$ Then we have that $t\{\!\{x := \vec{s}\}\!\} = \lambda^{\ell'} y.u$. We consider two further subcases, depending on whether t is an occurrence of the variable x.

3.2.1.1 **If** $t = x^\tau$ **for some type** τ**.** Then the list $\vec{s}$ must be of the form $[\lambda^{\ell'} y.u]$ where the external label of τ is precisely ℓ'. Then the situation is:

$$\mathtt{C}_1\langle (\lambda^\ell x.x^\tau)[\lambda^{\ell'} y.u]\,\vec{r}\rangle \to_\# \mathtt{C}_1\langle (\lambda^{\ell'} y.u)\,\vec{r}\rangle \to_\# \mathtt{C}_1\langle u\{\!\{y := \vec{r}\}\!\}\rangle$$

and we are in the situation of **Creation case I**.

3.2.1.2 **If** $t \neq x^\tau$ **for any type** τ**.** Then t must be an abstraction, namely $t = \lambda^{\ell'} y.u'$ where $u'\{\!\{x := \vec{s}\}\!\} = u$. The situation is:

$$\mathtt{C}_1\langle (\lambda^\ell x.\lambda^{\ell'} y.u')\,\vec{s}\,\vec{r}\rangle \to_\# \mathtt{C}_1\langle (\lambda^{\ell'} y.u'\{\!\{x := \vec{s}\}\!\})\,\vec{r}\rangle \to_\# \mathtt{C}_1\langle u'\{\!\{x := \vec{s}\}\!\}\{\!\{y := \vec{r}\}\!\}\rangle$$

and we are in the situation of **Creation case II**.

3.2.2 **Non-empty,** $\mathtt{C}_2' \neq \Box$ Then necessarily $\mathtt{C}_2'$ must be an abstraction, namely $\mathtt{C}_2' = \lambda^{\ell'} y.\mathtt{C}_2''$. The situation is:

$$\begin{array}{rcl} \mathtt{C}_1\langle (\lambda^{\ell'} y.\mathtt{C}_2''\langle (\lambda^\ell x.t)\,\vec{s}\rangle)\,\vec{r}\rangle & \to_\# & \mathtt{C}_1\langle (\lambda^{\ell'} y.\mathtt{C}_2''\langle t\{\!\{x := \vec{s}\}\!\}\rangle)\,\vec{r}\rangle \\ & \to_\# & \mathtt{C}_1\langle \mathtt{C}_2''\langle t\{\!\{x := \vec{s}\}\!\}\rangle\{\!\{y := \vec{r}\}\!\}\rangle \end{array}$$

Observe that S has an ancestor S_0, contradicting the hypothesis that R creates S. So this case is impossible.

3.3 **Right,** $\mathtt{C}_2 = (\lambda^{\ell'} y.u)[\vec{r}_1, \mathtt{C}_2', \vec{r}_2]$**.** The situation is:

$$\begin{array}{rcl} \mathtt{C}_1\langle (\lambda^{\ell'} y.u)\,[\vec{r}_1, \mathtt{C}_2'\langle (\lambda^\ell x.t)\vec{s}\rangle, \vec{r}_2]\rangle & \to_\# & \mathtt{C}_1\langle (\lambda^{\ell'} y.u)\,[\vec{r}_1, \mathtt{C}_2'\langle t\{\!\{x := \vec{s}\}\!\}\rangle, \vec{r}_2]\rangle \\ & \to_\# & \mathtt{C}_1\langle u\{\!\{y := [\vec{r}_1, \mathtt{C}_2'\langle t\{\!\{x := \vec{s}\}\!\}\rangle, \vec{r}_2]\}\!\}\rangle \end{array}$$

Observe that S has an ancestor S_0, contradicting the hypothesis that R creates S. So this case is impossible.

□

A.9 Proof of Lemma 3.19 — Basic Stability

We are to prove that if R, S are different coinitial steps such that R creates a step T, then R/S creates the step $T/(R/S)$.

An auxiliary lemma can be found at the end of the section

Proof. Let $R : \mathtt{C}\langle (\lambda^\ell x.t)\vec{s}\rangle \to_\# \mathtt{C}\langle t\{x := \vec{s}\}\rangle$, let $S \neq R$ be a step coinitial to R, and suppose that R creates a step T. We proceed by induction on the context $\mathtt{C}$ we argue that R/S creates $T/(S/R)$.

1. **Empty context,** $\mathtt{C} = \Box$**.** Then $R : (\lambda^\ell x.t)\vec{s} \to_\# t\{\!\{x := \vec{s}\}\!\}$. There are two cases for S, depending on whether it is internal to t or internal to one of the arguments of the list $\vec{s}$.

1.1 **If S is internal to t.** Then $t = \mathtt{C}_1\langle\Sigma\rangle$ where Σ is the redex contracted by S. Let Σ' denote the contractum of Σ. Then the situation is:

$$\begin{array}{ccccc}
(\lambda^\ell x.\mathtt{C}_1\langle\Sigma\rangle)\,\vec{s} & \xrightarrow{R} & \mathtt{C}_1\langle\Sigma\rangle\{\!\{x := \vec{s}\}\!\} & \xrightarrow{T} & \\
\Big\downarrow{\scriptstyle S} & & \Big\downarrow{\scriptstyle S/R} & & \\
(\lambda^\ell x.\mathtt{C}_1\langle\Sigma'\rangle)\,\vec{s} & \xrightarrow{R/S} & \mathtt{C}_1\langle\Sigma'\rangle\{\!\{x := \vec{s}\}\!\} & \xrightarrow{T/(S/R)} &
\end{array}$$

By Creation (Lemma 3.18), T must be created by case III, since there are no applications surrounding the subterm contracted by R. This means that $\mathtt{C}_1\langle\Sigma\rangle = \mathtt{C}_2\langle x^\tau\,\vec{u}\rangle$ where, moreover, $\vec{s}$ may be split as $[\vec{s}_1, \lambda^{\ell'} y.r, \vec{s}_2]$ in such a way that ℓ' is the external label of τ.
We consider three subcases, depending on whether the contexts $\mathtt{C}_1$ and $\mathtt{C}_2$ are disjoint, $\mathtt{C}_1$ is a prefix of $\mathtt{C}_2$, or $\mathtt{C}_2$ is a prefix of $\mathtt{C}_1$.

1.1.1 **If $\mathtt{C}_1$ and $\mathtt{C}_2$ are disjoint.** Then there is a two-hole context $\hat{\mathtt{C}}$ such that

$$\hat{\mathtt{C}}\langle\Box, x^\tau\vec{u}\rangle = \mathtt{C}_1 \qquad \hat{\mathtt{C}}\langle\Sigma, \Box\rangle = \mathtt{C}_2$$

Given any term, context, or list of terms $\mathtt{X}$ let us write $\mathtt{X}^*$ to denote $\mathtt{X}\{\!\{x := \vec{s}\}\!\}$. Then the situation is:

$$\begin{array}{ccccc}
(\lambda^\ell x.\hat{\mathtt{C}}\langle\Sigma, x^\tau\vec{u}\rangle)\,\vec{s} & \xrightarrow{R} & \hat{\mathtt{C}}^*\langle\Sigma^*, (\lambda^{\ell'} y.r)\vec{u}^*\rangle & \xrightarrow{T} & \hat{\mathtt{C}}^*\langle\Sigma^*, r\{\!\{y := \vec{u}\}\!\}\rangle \\
\Big\downarrow{\scriptstyle S} & & \Big\downarrow{\scriptstyle S/R} & & \\
(\lambda^\ell x.\hat{\mathtt{C}}\langle\Sigma', x^\tau\vec{u}\rangle)\,\vec{s} & \xrightarrow{R/S} & \hat{\mathtt{C}}^*\langle\Sigma'^*, (\lambda^{\ell'} y.r)\vec{u}^*\rangle & \xrightarrow{T/(S/R)} & \hat{\mathtt{C}}^*\langle\Sigma'^*, r\{\!\{y := \vec{u}^*\}\!\}\rangle
\end{array}$$

Then it is indeed the case that R/S creates $T/(S/R)$.

1.1.2 **If $\mathtt{C}_1$ is a prefix of $\mathtt{C}_2$.** Then $\mathtt{C}_2 = \mathtt{C}_1\langle\mathtt{C}_2'\rangle$ which means that $\Sigma = \mathtt{C}_2'\langle x^\tau\vec{u}\rangle$. Recall that Σ is a redex, so let us write $\Sigma = (\lambda^{\ell''} z.p)\vec{q}$. We consider two further subcases, depending on whether the hole of $\mathtt{C}_2'$ lies to the left or to the right of the application (observe that it cannot be at the root since $\lambda^{\ell''} z.p$ is not a variable).

- **If the hole of $\mathtt{C}_2'$ lies to the left of $(\lambda^{\ell''} z.p)\vec{q}$.** More precisely, we have that $\mathtt{C}_2' = (\lambda^{\ell''} z.\mathtt{C}_2'')\vec{q}$ and $p = \mathtt{C}_2''\langle x^\tau\vec{u}\rangle$. Given any term, context, or list of terms $\mathtt{X}$ let us write $\mathtt{X}^*$ to denote $\mathtt{X}\{\!\{x := \vec{s}\}\!\}$. Then the situation is:

$$\begin{array}{ccccc}
(\lambda^\ell x.\mathtt{C}_1\langle(\lambda^{\ell''} z.\mathtt{C}_2''\langle x^\tau\vec{u}\rangle)\vec{q}\rangle)\,\vec{s} & \xrightarrow{R} & \mathtt{C}_1^*\langle(\lambda^{\ell''} z.\mathtt{C}_2''^*\langle(\lambda^{\ell'} y.r)\vec{u}^*\rangle)\vec{q}^*\rangle & \xrightarrow{T} & \mathtt{C}_1^*\langle(\lambda^{\ell''} z.\mathtt{C}_2''^*\langle r\{\!\{y := \vec{u}^*\}\!\}\rangle)\vec{q}^*\rangle \\
\Big\downarrow{\scriptstyle S} & & \Big\downarrow{\scriptstyle S/R} & & \\
(\lambda^\ell x.\mathtt{C}_1\langle\mathtt{C}_2''\langle x^\tau\vec{u}\rangle\{\!\{z := \vec{q}\}\!\}\rangle)\,\vec{s} & \xrightarrow{R/S} & \mathtt{C}_1^*\langle\mathtt{C}_2''^*\langle(\lambda^{\ell'} y.r)\vec{u}^*\rangle\{\!\{z := \vec{q}^*\}\!\}\rangle & \xrightarrow{T/(S/R)} & \mathtt{C}_1^*\langle\mathtt{C}_2''^*\langle r\{\!\{y := \vec{u}^*\}\!\}\rangle\{\!\{z := \vec{q}^*\}\!\}\rangle
\end{array}$$

  and it is indeed the case that R/S creates $T/(S/R)$. Observe that we use the Substitution lemma (Lemma 2.14).
- **If the hole of $\mathtt{C}_2'$ lies to the right of $(\lambda^{\ell''} z.p)\vec{q}$.** More precisely, we have that $\mathtt{C}_2' = (\lambda^{\ell''} z.p)[\vec{q}_1, \mathtt{C}_2'', \vec{q}_2]$ and $\vec{q} = [\vec{q}_1, \mathtt{C}_2''\langle x^\tau\,\vec{u}\rangle, \vec{q}_2]$. Given any term, context, or list of terms $\mathtt{X}$ let us write $\mathtt{X}^*$ to denote $\mathtt{X}\{\!\{x := \vec{s}\}\!\}$.
  Moreover, by the parameter/argument correspondence (Lemma A.9) there is exactly one free occurrence of z in p whose type coincides with the type

of $\mathtt{C}_2''\langle x^\tau\,\vec{u}\rangle$. Let $p = \mathtt{C}_3\langle z^\sigma\rangle$ where the hole of $\mathtt{C}_3$ marks the position of such occurrence. Observe that $\mathtt{C}_3\langle z^\sigma\rangle\{\!\{z := \vec{q}\}\!\} = \mathtt{C}_3\{\!\{z := \vec{q}\}\!\}\langle\mathtt{C}_2''\langle x^\tau\,\vec{u}\rangle\rangle$. Observe that:

$$\begin{array}{rcl}\mathtt{C}_3\langle z^\sigma\rangle\{\!\{z := \vec{q}^*\}\!\} &=& \mathtt{C}_3\{\!\{z := \vec{q}^*\}\!\}\langle(\mathtt{C}_2''\langle x^\tau\,\vec{u}\rangle)^*\rangle \\ &=& \mathtt{C}_3\{\!\{z := \vec{q}^*\}\!\}\langle\mathtt{C}_2''^*\langle(\lambda^{\ell'}y.r)\,\vec{u}^*\rangle\rangle\end{array}$$

Moreover, if we write $\mathtt{C}_3'$ for the context $\mathtt{C}_3\{\!\{z := \vec{q}\}\!\}$, then we have that

$$\begin{array}{rcll}(\mathtt{C}_3')^* &=& (\mathtt{C}_3\{\!\{z := \vec{q}\}\!\})^* & \\ &=& \mathtt{C}_3^*\{\!\{z := \vec{q}^*\}\!\} & \text{by the Substitution lemma (Lemma 2.14)}\end{array}$$

Then the situation is as follows:

$$\begin{array}{ccccc}(\lambda^\ell x.\mathtt{C}_1\langle(\lambda^{\ell''}z.\mathtt{C}_3\langle z^\sigma\rangle)\vec{q}\rangle)\,\vec{s} & \xrightarrow{R} & \mathtt{C}_1^*\langle(\lambda^{\ell''}z.\mathtt{C}_3^*\langle z^\sigma\rangle)\vec{q}^*\rangle & \xrightarrow{T} & \\ \Big\downarrow{\scriptstyle S} & & \Big\downarrow{\scriptstyle S/R} & & \\ (\lambda^\ell x.\mathtt{C}_1\langle\mathtt{C}_3'\langle\mathtt{C}_2''\langle x^\tau\,\vec{u}\rangle\rangle\rangle)\,\vec{s} & \xrightarrow{R/S} & \mathtt{C}_1^*\langle(\mathtt{C}_3')^*\langle\mathtt{C}_2''^*\langle(\lambda^{\ell'}y.r)\,\vec{u}^*\rangle\rangle\rangle & \xrightarrow{T/(S/R)} & \end{array}$$

Observe that the names of the steps T and $T/(S/R)$ are both ℓ', which means that indeed R/S creates $T/(S/R)$.

1.1.3 **If $\mathtt{C}_2$ is a prefix of $\mathtt{C}_1$.** Then $\mathtt{C}_1 = \mathtt{C}_2\langle\mathtt{C}_1'\rangle$ which means $x^\tau\,\vec{u} = \mathtt{C}_1'\langle\Sigma\rangle$. Since Σ is a redex, $\mathtt{C}_1'$ must be of the form $x^\tau\,[\vec{u}_1, \mathtt{C}_1'', \vec{u}_2]$ and $\vec{u} = [\vec{u}_1, \mathtt{C}_1''\langle\Sigma\rangle, \vec{u}_2]$. Given any term, context, or list of terms $\mathtt{X}$ let us write $\mathtt{X}^*$ to denote $\mathtt{X}\{\!\{x := \vec{s}\}\!\}$. Then the situation is:

$$\begin{array}{ccccc}(\lambda^\ell x.\mathtt{C}_2\langle x^\tau[\vec{u}_1,\mathtt{C}_1''\langle\Sigma\rangle,\vec{u}_2]\rangle)\vec{s} & \xrightarrow{R} & \mathtt{C}_2^*\langle(\lambda^{\ell'}y.r)[\vec{u}_1^*,\mathtt{C}_1''^*\langle\Sigma^*\rangle,\vec{u}_2^*]\rangle & \xrightarrow{T} & \mathtt{C}_2^*\langle r\{\!\{y := [\vec{u}_1^*,\mathtt{C}_1''^*\langle\Sigma^*\rangle,\vec{u}_2^*]\}\!\}\rangle \\ \Big\downarrow{\scriptstyle S} & & \Big\downarrow{\scriptstyle S/R} & & \\ (\lambda^\ell x.\mathtt{C}_2\langle x^\tau[\vec{u}_1,\mathtt{C}_1''\langle\Sigma'\rangle,\vec{u}_2]\rangle)\vec{s} & \xrightarrow{R/S} & \mathtt{C}_2^*\langle(\lambda^{\ell'}y.r)[\vec{u}_1^*,\mathtt{C}_1''^*\langle\Sigma'^*\rangle,\vec{u}_2^*]\rangle & \xrightarrow{T/(S/R)} & \mathtt{C}_2^*\langle r\{\!\{y := [\vec{u}_1^*,\mathtt{C}_1''^*\langle\Sigma'^*\rangle,\vec{u}_2^*]\}\!\}\rangle\end{array}$$

Note that indeed R/S creates $T/(S/R)$.

1.2 **If S is internal to $\vec{s}$.** Then $\vec{s} = [\vec{s}_1, \mathtt{C}_1\langle\Sigma\rangle, \vec{s}_2]$ where Σ is the redex contracted by S. Let Σ' denote the contractum of Σ. Then the situation is:

$$\begin{array}{ccccc}(\lambda^\ell x.t)\,[\vec{s}_1,\mathtt{C}_1\langle\Sigma\rangle,\vec{s}_2] & \xrightarrow{R} & t\{\!\{x := [\vec{s}_1,\mathtt{C}_1\langle\Sigma\rangle,\vec{s}_2]\}\!\} & \xrightarrow{T} & \\ \Big\downarrow{\scriptstyle S} & & \Big\downarrow{\scriptstyle S/R} & & \\ (\lambda^\ell x.t)\,[\vec{s}_1,\mathtt{C}_1\langle\Sigma'\rangle,\vec{s}_2] & \xrightarrow{R/S} & t\{\!\{x := [\vec{s}_1,\mathtt{C}_1\langle\Sigma'\rangle,\vec{s}_2]\}\!\} & \xrightarrow{T/(S/R)} & \end{array}$$

By Creation (Lemma 3.18), T must be created by case III, since there are no applications surrounding the subterm contracted by R. This means that $t = \mathtt{C}_2\langle x^\tau\,\vec{u}\rangle$, and there is a term in the list $[\vec{s}_1, \mathtt{C}_1\langle\Sigma\rangle, \vec{s}_2]$ of the form $\lambda^{\ell'}y.r$ such that ℓ' is also the external label of the type τ. There are two subcases, depending on whether such term is $\mathtt{C}_1\langle\Sigma\rangle$ or a different term in the list $[\vec{s}_1, \mathtt{C}_1\langle\Sigma\rangle, \vec{s}_2]$.

1.2.1 **If $\lambda^{\ell'}y.r = \mathtt{C}_1\langle\Sigma\rangle$.** Note that $\mathtt{C}_1$ cannot be empty, since this would imply that $\lambda^{\ell'}y.r = \Sigma$; however Σ is a redex, and in particular an application, so this cannot be the case. Hence the context $\mathtt{C}_1$ must be non-empty, *i.e.* $\mathtt{C}_1 = \lambda^{\ell'}y.\mathtt{C}_1'$ with

$r = \mathtt{C}'_1\langle\Sigma\rangle$. Given any term, context, or list of terms $\mathtt{X}$ let us write $\mathtt{X}^*$ to denote $\mathtt{X}\{\!\{x := [\vec{s}_1, \vec{s}_2]\}\!\}$. Then the situation is:

$$\begin{array}{ccccc}
(\lambda^\ell x.\mathtt{C}_2\langle x^\tau\,\vec{u}\rangle)[\vec{s}_1, \lambda^{\ell'} y.\mathtt{C}'_1\langle\Sigma\rangle, \vec{s}_2] & \xrightarrow{R} & \mathtt{C}^*_2\langle(\lambda^{\ell'} y.\mathtt{C}'_1\langle\Sigma\rangle)\,\vec{u}^*\rangle & \xrightarrow{T} & \\
\downarrow{\scriptstyle S} & & \downarrow{\scriptstyle S/R} & & \\
(\lambda^\ell x.\mathtt{C}_2\langle x^\tau\,\vec{u}\rangle)[\vec{s}_1, \lambda^{\ell'} y.\mathtt{C}'_1\langle\Sigma'\rangle, \vec{s}_2] & \xrightarrow{R/S} & \mathtt{C}^*_2\langle(\lambda^{\ell'} y.\mathtt{C}'_1\langle\Sigma'\rangle)\,\vec{u}^*\rangle & \xrightarrow{T/(S/R)} &
\end{array}$$

Note that indeed R/S creates $T/(S/R)$.

1.2.2 **If $\lambda^{\ell'} y.r \neq \mathtt{C}_1\langle\Sigma\rangle$.** Then $\lambda^{\ell'} y.r$ is either one of the terms in the list $\vec{s}_1$ or one of the terms in the list $\vec{s}_2$. Given any term, context, or list of terms $\mathtt{X}$ let $\mathtt{X}^*$ denote $\mathtt{X}\{\!\{x := [\vec{s}_1, \mathtt{C}_1\langle\Sigma\rangle\vec{s}_2]\}\!\}$ and let $\mathtt{X}^\dagger$ denote $\mathtt{X}\{\!\{x := [\vec{s}_1, \mathtt{C}_1\langle\Sigma'\rangle\vec{s}_2]\}\!\}$. The situation is as follows:

$$\begin{array}{ccccc}
(\lambda^\ell x.\mathtt{C}_2\langle x^\tau\,\vec{u}\rangle)\,[\vec{s}_1, \mathtt{C}_1\langle\Sigma\rangle, \vec{s}_2] & \xrightarrow{R} & \mathtt{C}^*_2\langle(\lambda^{\ell'} y.r)\,\vec{u}^*\rangle & \xrightarrow{T} & \mathtt{C}^*_2\langle r\{\!\{y := \vec{u}^*\}\!\}\rangle \\
\downarrow{\scriptstyle S} & & \downarrow{\scriptstyle S/R} & & \\
(\lambda^\ell x.\mathtt{C}_2\langle x^\tau\,\vec{u}\rangle)\,[\vec{s}_1, \mathtt{C}_1\langle\Sigma'\rangle, \vec{s}_2] & \xrightarrow{R/S} & \mathtt{C}^\dagger_2\langle(\lambda^{\ell'} y.r)\,\vec{u}^\dagger\rangle & \xrightarrow{T/(S/R)} & \mathtt{C}^\dagger_2\langle r\{\!\{y := \vec{u}^\dagger\}\!\}\rangle
\end{array}$$

Note that indeed R/S creates $T/(S/R)$.

2. **Under an abstraction, $\mathtt{C} = \lambda^\ell y.\mathtt{C}'$.** Let Δ be the redex contracted by R, and let Δ' denote its contractum. The step R is of the form $\lambda^\ell y.\mathtt{C}'[\Delta] \to_\# \lambda^\ell y.\mathtt{C}'[\Delta']$. The steps S and T cannot be located at the root of the term since the root of the term is not an application. Hence the positions of S and T are necessarily internal to the outermost abstraction, *i.e.* the situation is as follows:

$$\begin{array}{ccccc}
\lambda^\ell y.\mathtt{C}'[\Delta] & \xrightarrow{R} & \lambda^\ell y.\mathtt{C}'[\Delta'] & \xrightarrow{T} & \lambda^\ell y.s \\
\downarrow{\scriptstyle S} & & \downarrow{\scriptstyle S/R} & & \\
\lambda^\ell y.u & \xrightarrow{R/S} & \lambda^\ell y.u' & \xrightarrow{T/(S/R)} & \lambda^\ell y.u''
\end{array}$$

Consider the diagram without the outermost abstraction, where R', S', and T' are the steps isomorphic to R, S, and T respectively:

$$\begin{array}{ccccc}
\mathtt{C}'[\Delta] & \xrightarrow{R'} & \mathtt{C}'[\Delta'] & \xrightarrow{T'} & s \\
\downarrow{\scriptstyle S'} & & \downarrow{\scriptstyle S'/R'} & & \\
u & \xrightarrow{R'/S'} & u' & \xrightarrow{T'/(S'/R')} & u''
\end{array}$$

Note that R' creates T', so by *i.h.* R'/S' creates $T'/(S'/R')$. Hence we conclude that R/S creates $T/(S/R)$, as required.

3. **Left of an application, $\mathtt{C} = \mathtt{C}'\vec{u}$.** Let Δ be the redex contracted by R, and let Δ' denote its contractum. The step R is of the form $\mathtt{C}'\langle\Delta\rangle\vec{u} \to_\# \mathtt{C}'\langle\Delta'\rangle\vec{u}$. We consider three subcases, according to Creation (Lemma 3.18), depending on whether T is created by the creation case I, II, or III.

3.1 **Creation case I.** Then $\mathtt{C}'$ is empty and Δ has the following particular shape: $\Delta = (\lambda^\ell x.x^\tau)[\lambda^{\ell'} y.r]$. The step S can be either internal to the subterm r, or internal to one

of the subterms in the list $\vec{u}$. If S is internal to r, let r' denote the term that results from r after the step S. Then the situation is:

$$\begin{array}{ccccc}
(\lambda^{\ell} x.x^{\tau})[\lambda^{\ell'} y.r]\,\vec{u} & \xrightarrow{R} & (\lambda^{\ell'} y.r)\,\vec{u} & \xrightarrow{\quad T \quad} & r\{\!\{y := \vec{u}\}\!\} \\
\Big\downarrow{\scriptstyle S} & & \Big\downarrow{\scriptstyle S/R} & & \\
(\lambda^{\ell} x.x^{\tau})[\lambda^{\ell'} y.r']\,\vec{u} & \xrightarrow{R/S} & (\lambda^{\ell'} y.r')\,\vec{u} & \xrightarrow{\quad T/(S/R) \quad} & r'\{\!\{y := \vec{u}\}\!\}
\end{array}$$

Note that indeed R/S creates $T/(S/R)$, as required. If, on the other hand, S is internal to $\vec{u}$, the situation is similar.

3.2 **Creation case II.** Then $\mathtt{C}'$ is empty and Δ has the following particular shape: $\Delta = (\lambda^{\ell} x.\lambda^{\ell'} y.r)\,\vec{s}$. The step S can be either internal to r, internal to $\vec{s}$, or internal to $\vec{u}$. If S is internal to r, let r' denote the term that results from r after the step S. Then the situation is:

$$\begin{array}{ccccc}
(\lambda^{\ell} x.\lambda^{\ell'} y.r)\,\vec{s}\,\vec{u} & \xrightarrow{R} & (\lambda^{\ell'} y.r\{\!\{x := \vec{s}\}\!\})\,\vec{u} & \xrightarrow{T} & r\{\!\{x := \vec{s}\}\!\}\{\!\{y := \vec{u}\}\!\} \\
\Big\downarrow{\scriptstyle S} & & \Big\downarrow{\scriptstyle S/R} & & \\
(\lambda^{\ell} x.\lambda^{\ell'} y.r')\,\vec{s}\,\vec{u} & \xrightarrow{R/S} & (\lambda^{\ell'} y.r'\{\!\{x := \vec{s}\}\!\})\,\vec{u} & \xrightarrow{T/(S/R)} & r'\{\!\{x := \vec{s}\}\!\}\{\!\{y := \vec{u}\}\!\}
\end{array}$$

Note that indeed R/S creates $T/(S/R)$, as required. If, on the other hand, S is internal to $\vec{s}$ or $\vec{u}$, the situation is similar.

3.3 **Creation case III.** Recall that the step R is of the form $R : \mathtt{C}'\langle\Delta\rangle\,\vec{u} \to_{\#} \mathtt{C}'\langle\Delta'\rangle\,\vec{u}$. Since the step T is created by creation case III, it does not take place at the root of $\mathtt{C}'[\Delta']\,\vec{u}$, but rather it is internal to $\mathtt{C}'[\Delta']$. We consider three subcases, depending on whether the step S takes place at the root, to the left of the application or to the right of the application.

3.3.1 **If S takes place at the root.** Then $\mathtt{C}'\langle\Delta\rangle$ is an abstraction, so $\mathtt{C}'$ cannot be empty, *i.e.* we have that $\mathtt{C}' = \lambda^{\ell'} y.\mathtt{C}''$. Moreover, since T is created by Creation case III, we have that Δ has the following specific shape: $\Delta = (\lambda^{\ell} x.\mathtt{C}_2\langle x^{\tau}\,\vec{r}\rangle)\,\vec{s}$ where $\vec{s} = [\vec{s}_1, \lambda^{\ell''} z.p, \vec{s}_2]$ and such that ℓ'' is the external label of τ.
Given any term, context, or list of terms $\mathtt{X}$ let $\mathtt{X}^*$ denote $\mathtt{X}\{\!\{x := \vec{s}\}\!\}$ and let $\mathtt{X}^{\dagger}$ denote $\mathtt{X}\{\!\{y := \vec{u}\}\!\}$. Note also that by the Substitution lemma (Lemma 2.14) we have that $\mathtt{X}^{\dagger}\{\!\{x := \vec{s}^{\dagger}\}\!\} = \mathtt{X}^{*\dagger}$. Then the situation is the following:

$$\begin{array}{ccccc}
(\lambda^{\ell'} y.\mathtt{C}''\langle(\lambda^{\ell} x.\mathtt{C}_2\langle x^{\tau}\,\vec{r}\rangle)\,\vec{s}\rangle)\,\vec{u} & \overset{R}{\succ} & (\lambda^{\ell'} y.\mathtt{C}''\langle\mathtt{C}_2^*\langle(\lambda^{\ell''} z.p)\,\vec{r}^*\rangle\rangle)\,\vec{u} & \overset{T}{\succ} & (\lambda^{\ell'} y.\mathtt{C}''\langle\mathtt{C}_2^*\langle p\{\!\{z := \vec{r}^*\}\!\}\rangle\rangle)\,\vec{u} \\
\Big\downarrow{\scriptstyle S} & & \Big\downarrow{\scriptstyle S/R} & & \\
\mathtt{C}''^{\dagger}\langle(\lambda^{\ell} x.\mathtt{C}_2^{\dagger}\langle x^{\tau}\,\vec{r}^{\dagger}\rangle)\,\vec{s}^{\dagger}\rangle & \xrightarrow{R/S} & \mathtt{C}''^{\dagger}\langle\mathtt{C}_2^{*\dagger}\langle(\lambda^{\ell''} z.p^{\dagger})\,\vec{r}^{*\dagger}\rangle\rangle & \xrightarrow{T/(S/R)} & \mathtt{C}''^{\dagger}\langle\mathtt{C}_2^{*\dagger}\langle p^{\dagger}\{\!\{z := \vec{r}^{*\dagger}\}\!\}\rangle\rangle
\end{array}$$

Note that, indeed, R/S creates $T/(S/R)$.

3.3.2 **If S is internal to $\mathtt{C}'\langle\Delta\rangle$.** Observe that in such case all three steps R, S, and T are internal to the left of the application. Then situation is:

$$\begin{array}{ccccc}
\mathtt{C}'\langle\Delta\rangle\,\vec{u} & \xrightarrow{R} & \mathtt{C}'\langle\Delta'\rangle\,\vec{u} & \xrightarrow{\quad T \quad} & r\,\vec{u} \\
\Big\downarrow{\scriptstyle S} & & \Big\downarrow{\scriptstyle S/R} & & \\
p\,\vec{u} & \xrightarrow{R/S} & p'\,\vec{u} & \xrightarrow{\quad T/(S/R) \quad} & p''\,\vec{u}
\end{array}$$

and we may conclude by *i.h.*, similarly as in item 2 of this lemma.

3.3.3 **If S is internal to $\vec{u}$.** Then the situation is:

$$\begin{array}{ccccc} \mathtt{C}'\langle\Delta\rangle\,\vec{u} & \xrightarrow{R} & \mathtt{C}'\langle\Delta'\rangle\,\vec{u} & \xrightarrow{T} & p\,\vec{u} \\ \downarrow{\scriptstyle S} & & \downarrow{\scriptstyle S/R} & & \\ \mathtt{C}'\langle\Delta\rangle\,\vec{r} & \xrightarrow{R/S} & \mathtt{C}'\langle\Delta'\rangle\,\vec{r} & \xrightarrow{T/(S/R)} & p\,\vec{r} \end{array}$$

It is immediate to note in this case that R/S creates $T/(S/R)$, since the two-step sequences RT and $(R/S)(T/(S/R))$ are both isomorphic to $\mathtt{C}'\langle\Delta\rangle \to_{\#} \mathtt{C}'\langle\Delta'\rangle \to_{\#} p$, only going below different contexts.

4. **Right of an application,** $\mathtt{C} = u[\vec{r}_1, \mathtt{C}', \vec{r}_2]$**.** Let Δ be the redex contracted by R, and let Δ' denote its contractum. The step R is of the form $u[\vec{r}_1, \mathtt{C}'\langle\Delta\rangle, \vec{r}_2] \to_{\#} u[\vec{r}_1, \mathtt{C}'\langle\Delta'\rangle, \vec{r}_2]$. We consider three subcases, depending on whether the step S takes place at the root, to the left of the application, or to the right of the application.

4.1 **If S takes place at the root.** Then u is an abstraction $u = \lambda^{\ell'} y.u'$. Moreover, by the parameter/argument correspondence (Lemma A.9), there is a free occurrence of y in u' whose type is also the type of the argument $\mathtt{C}'\langle\Delta\rangle$. Let us write u' as $u' = \mathtt{C}_1\langle y^{\tau}\rangle$, where the hole of $\mathtt{C}_1$ marks the position of such occurrence. Given any term, context, or list of terms $\mathtt{X}$ let $\mathtt{X}^*$ denote $\mathtt{X}\{\!\{x := \vec{r}_1, \vec{r}_2\}\!\}$. Then we have that the situation is:

$$\begin{array}{ccccc} (\lambda^{\ell'} y.\mathtt{C}_1\langle y^{\tau}\rangle)[\vec{r}_1, \mathtt{C}'\langle\Delta\rangle, \vec{r}_2] & \xrightarrow{R} & (\lambda^{\ell'} y.\mathtt{C}_1\langle y^{\tau}\rangle)[\vec{r}_1, \mathtt{C}'\langle\Delta'\rangle, \vec{r}_2] & \xrightarrow{T} & (\lambda^{\ell'} y.\mathtt{C}_1\langle y^{\tau}\rangle)[\vec{r}_1, p, \vec{r}_2] \\ \downarrow{\scriptstyle S} & & \downarrow{\scriptstyle S/R} & & \\ \mathtt{C}_1^*\langle\mathtt{C}'\langle\Delta\rangle\rangle & \xrightarrow{R/S} & \mathtt{C}_1^*\langle\mathtt{C}'\langle\Delta'\rangle\rangle & \xrightarrow{T/(S/R)} & \mathtt{C}_1^*\langle p\rangle \end{array}$$

It is then immediate to note that R/S creates $T/(S/R)$, since the two-step sequences RT and $(R/S)(T/(S/R))$ are both isomorphic to $\mathtt{C}'\langle\Delta\rangle \to_{\#} \mathtt{C}'\langle\Delta'\rangle \to_{\#} p$, only going below different contexts.

4.2 **If S is internal to the left of the application.** Then the situation is:

$$\begin{array}{ccccc} u[\vec{r}_1, \mathtt{C}'\langle\Delta\rangle, \vec{r}_2] & \xrightarrow{R} & u[\vec{r}_1, \mathtt{C}'\langle\Delta'\rangle, \vec{r}_2] & \xrightarrow{T} & u[\vec{r}_1, p, \vec{r}_2] \\ \downarrow{\scriptstyle S} & & \downarrow{\scriptstyle S/R} & & \\ u'[\vec{r}_1, \mathtt{C}'\langle\Delta\rangle, \vec{r}_2] & \xrightarrow{R/S} & u'[\vec{r}_1, \mathtt{C}'\langle\Delta'\rangle, \vec{r}_2] & \xrightarrow{T/(S/R)} & u'[\vec{r}_1, p, \vec{r}_2] \end{array}$$

Then since RT and $(R/S)(T/(S/R))$ are isomorphic, it is immediate to conclude, similarly as in item 3.3.3 of this lemma.

4.3 **If S is internal to the right of the application.** We consider two further subcases, depending on whether S is internal to the argument $\mathtt{C}'\langle\Delta\rangle$, or otherwise.

4.3.1 **If S is internal to the argument $\mathtt{C}'\langle\Delta\rangle$.** Then the situation is:

$$\begin{array}{ccccc} u[\vec{r}_1, \mathtt{C}'\langle\Delta\rangle, \vec{r}_2] & \xrightarrow{R} & u[\vec{r}_1, \mathtt{C}'\langle\Delta'\rangle, \vec{r}_2] & \xrightarrow{T} & u[\vec{r}_1, p, \vec{r}_2] \\ \downarrow{\scriptstyle S} & & \downarrow{\scriptstyle S/R} & & \\ u[\vec{r}_1, q, \vec{r}_2] & \xrightarrow{R/S} & u[\vec{r}_1, q', \vec{r}_2] & \xrightarrow{T/(S/R)} & u[\vec{r}_1, q'', \vec{r}_2] \end{array}$$

and we may conclude by *i.h.*, similarly as in item 2 of this lemma.

4.3.2 **If S is not internal to the argument $\mathtt{C}'\langle\Delta\rangle$.** Then it is either internal to $\vec{r}_1$ or to $\vec{r}_2$. If it is internal to $\vec{r}_1$, the situation is:

$$\begin{array}{ccccc} u[\vec{r}_1, \mathtt{C}'\langle\Delta\rangle, \vec{r}_2] & \xrightarrow{R} & u[\vec{r}_1, \mathtt{C}'\langle\Delta'\rangle, \vec{r}_2] & \xrightarrow{T} & u[\vec{r}_1, p, \vec{r}_2] \\ \Big\downarrow{\scriptstyle S} & & \Big\downarrow{\scriptstyle S/R} & & \\ u[\vec{r}'_1, \mathtt{C}'\langle\Delta\rangle, \vec{r}_2] & \xrightarrow{R/S} & u[\vec{r}'_1, \mathtt{C}'\langle\Delta'\rangle, \vec{r}_2] & \xrightarrow{T/(S/R)} & u[\vec{r}'_1, p, \vec{r}_2] \end{array}$$

Then since RT and $(R/S)(T/(S/R))$ are isomorphic, it is immediate to conclude, similarly as in item 3.3.3 of this lemma.

□

Lemma A.9 (Parameter/argument correspondence)**.** *If $(\lambda^\ell x.t)[s_1, \ldots, s_n]$ is a correct term then there are exactly n free occurrences of x in t. More precisely, t can be written as $\hat{\mathtt{C}}\langle x^{\tau_1}, \ldots, x^{\tau_n}\rangle$ where $\hat{\mathtt{C}}$ is an n-hole context and type τ_i is the type of s_i for all $1 \leq i \leq n$.*

Proof. Consider the (unique) type derivation for $(\lambda^\ell x.t)[s_1, \ldots, s_n]$. The last rule must be:

$$\dfrac{\dfrac{\Gamma, x : [\sigma_1, \ldots, \sigma_n] \vdash t : \tau}{\Gamma \vdash \lambda^\ell x.t : [\sigma_1, \ldots, \sigma_n] \xrightarrow{\ell} \tau} \to_{\mathrm{I}} \qquad \Delta_i \vdash s_i : \sigma_i \quad \text{for all } i = 1..n}{\Gamma +_{i=1}^{n} \Delta_i \vdash (\lambda^\ell x.t)[s_1, \ldots, s_n] : \tau} \to_{\mathrm{E}}$$

It is then easy to conclude by resorting to Linearity Lemma 2.17. □

A.10 Auxiliary lemmas for Section 3.4 — Lattices and Derivation Spaces

Below the reader can find some auxiliary lemmas used in the proofs of the main results in Section 3.4.

Lemma A.10 (Projections are decreasing)**.** *Let $R \in \rho$. Then $|\rho| = 1 + |\rho/R|$.*

Proof. Observe that $R \sqsubseteq \rho$ by Lemma 3.11. So $\rho \equiv R(\rho/R)$, which gives us that:

$$\begin{array}{rcll} |\rho| & = & \#\mathsf{names}(\rho) & \text{by Corollary 3.14} \\ & = & \#\mathsf{names}(R(\rho/R)) & \text{by Corollary 3.17, since } \rho \equiv R(\rho/R) \\ & = & \#(\mathsf{names}(R) \uplus \mathsf{names}(\rho/R) & \text{by Lemma 3.13} \\ & = & 1 + \#\mathsf{names}(\rho/R) & \\ & = & 1 + |\rho/R| & \text{by Corollary 3.14} \end{array}$$

□

Lemma A.11 (Properties of disjoint derivations)**.** *Let ρ, σ be coinitial derivations. Then the following are equivalent:*

1. $\mathsf{names}(\rho) \cap \mathsf{names}(\sigma) = \varnothing$.
2. $\rho \sqcap \sigma = \epsilon$.
3. *There is no step common to ρ and σ.*

In this case we say that ρ and σ are **disjoint**.

Proof. The implication $(1 \implies 2)$ is immediate since if we suppose that $\rho \sqcap \sigma$ is non-empty then the first step of $\rho \sqcap \sigma$ is a step T such that $T \in \rho$ and $T \in \sigma$. By Characterization of belonging (Lemma 3.11), this means that $\mathsf{name}(T) \in \mathsf{names}(\rho) \cap \mathsf{names}(\sigma)$, contradicting the fact that $\mathsf{name}(T)$ and $\mathsf{names}(\rho)$ are disjoint.

The implication $(2 \implies 3)$ is immediate by definition of $\rho \sqcap \sigma$.

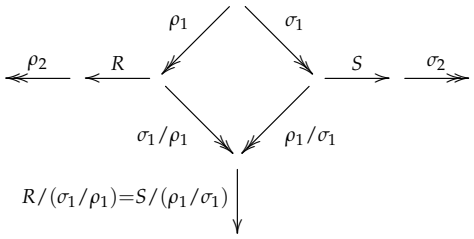
Let us check that the implication $(3 \implies 1)$ holds. By the contrapositive, suppose that $\mathsf{names}(\rho)$ and $\mathsf{names}(\sigma)$ are not disjoint, and let us show that there is a step common to ρ and σ. Since $\mathsf{names}(\rho) \cap \mathsf{names}(\sigma) \neq \varnothing$, we know that the derivation ρ can be written as $\rho = \rho_1 R \rho_2$ where $\mathsf{name}(R) \in \mathsf{names}(\sigma)$. Without loss of generality we may suppose that R is the first step in ρ with that property, *i.e.* that $\mathsf{names}(\rho_1) \cap \mathsf{names}(\sigma) = \varnothing$. Moreover, let us write σ as $\sigma = \sigma_1 S \sigma_2$ where $\mathsf{name}(R) = \mathsf{name}(S)$.

Observe that the name of R does not appear anywhere along the sequence of steps ρ_1, *i.e.* that $\mathsf{name}(R) \notin \mathsf{names}(\rho_1)$, as a consequence of the fact that no names are ever repeated in any sequence of steps (Lemma 3.13). This implies that $\mathsf{name}(S) \notin \mathsf{names}(\rho_1/\sigma_1)$. Indeed:

$$\mathsf{name}(S) = \mathsf{name}(R) \notin \mathsf{names}(\rho_1) \supseteq \mathsf{names}(\rho_1) \setminus \mathsf{names}(\sigma_1) =^{(\text{Lemma } 3.15)} \mathsf{names}(\rho_1/\sigma_1)$$

This means that S is not erased by the derivation ρ_1/σ_1. More precisely, $S/(\rho_1/\sigma_1)$ is a singleton.

Symmetrically, $R/(\sigma_1/\rho_1)$ is a singleton. Moreover, $\mathsf{name}(S/(\rho_1/\sigma_1)) = \mathsf{name}(S) = \mathsf{name}(R) = \mathsf{name}(R/(\sigma_1/\rho_1))$ so we have that $S/(\rho_1/\sigma_1) = R/(\sigma_1/\rho_1)$. The situation is the following, where $\mathsf{names}(\rho_1) \cap \mathsf{names}(\sigma_1) = \varnothing$:



By Full stability (Lemma 3.20) this means that there exists a step T such that $T/\rho_1 = R$ and $T/\sigma_1 = S$. Then $T \in \rho_1 R \rho_2 = \rho$ and also $T \in \sigma_1 S \sigma_2 = \sigma$ so T is common to ρ and σ, by which we conclude. □

Lemma A.12. *Let ρ and σ be coinitial derivations. Then* $\mathsf{names}(\rho \sqcap \sigma) \subseteq \mathsf{names}(\rho)$.

Proof. By induction on the length of $\rho \cap \sigma$:

1. **Empty, $\rho \sqcap \sigma = \epsilon$.** Then $\mathsf{names}(\rho \cap \sigma) = \varnothing \subseteq \mathsf{names}(\rho)$ is immediate.
2. **Non-empty, $\rho \sqcap \sigma = T(\rho/T \sqcap \sigma/T)$, where T is a step common to ρ and σ.** Then since T is common to ρ and σ, we have that $\mathsf{name}(T) \in \mathsf{names}(\rho)$. Moreover, by *i.h.* $\mathsf{names}(\rho/T \sqcap \sigma/T) \subseteq \mathsf{names}(\rho/T)$. So:

$$\begin{array}{rcll} \mathsf{names}(\rho \cap \sigma) & = & \{\mathsf{name}(T)\} \cup \mathsf{names}(\rho/T \sqcap \sigma/T) & \\ & \subseteq & \mathsf{names}(\rho) \cup \mathsf{names}(\rho/T) & \\ & = & \mathsf{names}(\rho) \cup (\mathsf{names}(\rho) \setminus \{\mathsf{name}(T)\} & \text{by Lemma 3.15} \\ & = & \mathsf{names}(\rho) & \end{array}$$

as required.

□

A.11 Auxiliary lemmas for Section 4.1 — Refinements

Lemma A.13 (Refinement of a substitution)**.** *If t' is of the form $t'\{x := [s'_1, \dots, s'_n]\}$, where $t' \ltimes t$ and $s'_i \ltimes s$ for all $1 \leq i \leq n$ then $t' \ltimes t\{x := s\}$.*

Proof. By induction on t.

1. **Variable (same),** $t = x$**.** In this case, $t' = x$. Then, $[s'_i]_{i=1}^n = [s']$ and $x\{x := [s']\} = s'$, which by hypothesis refines $s = x\{x := s\}$.
2. **Variable (different),** $t = y$**.** In this case $t' = y$. Then, $[s'_i]_{i=1}^n = [\,]$ and $y\{x := [\,]\} = y$, which refines $y = y\{x := s\}$.
3. **Abstraction,** $t = \lambda y.r$**.** In this case $t' = \lambda^\ell y.r'$, where $r \rtimes r'$.
   Also $(\lambda y.r)\{x := s\} = \lambda y.r\{x := s\}$, and $(\lambda^\ell y.r')\{x := [s'_1, \dots, s'_n]\} = \lambda^\ell y.r'\{x := [s'_1, \dots, s'_n]\}$. By inductive hypothesis $r'\{x := [s'_1, \dots, s'_n]\} \ltimes r\{x := s\}$, so we are done.
4. **Application,** $t = ru$**.** In this case $t' = r'[u'_1, \dots, u'_n]$, where $r \rtimes r'$ and $u \rtimes u'_i$.
   Also, $(ru)\{x := s\} = r\{x := s\}u\{x := s\}$ and $(r'[u'_1, \dots, u'_n])\{x := \vec{s}\} = r'\{x := \vec{s}_0\}u_i\{x := \vec{s}_i\}$.
   But by inductive hypothesis $r\{x := s\} \rtimes r'\{x := \vec{s}_0\}$ and $u\{x := s\} \rtimes u_i\{x := \vec{s}_i\}$ for all $i \in \{1, \dots, n\}$, so we are done.

□

Lemma A.14 (Refinement of a context)**.** *The following are equivalent, which relates refinement and contexts:*

1. *$t' \ltimes \mathtt{C}\langle s\rangle$,*
2. *t' is of the form $\mathtt{C}'\langle s'_1, \dots, s'_n\rangle$, where $\mathtt{C}'$ is an n-hole context such that $\mathtt{C}' \ltimes \mathtt{C}$ and $s'_i \ltimes s$ for all $1 \leq i \leq n$. Note that n might be 0, in which case $\mathtt{C}'$ is a term.*

Moreover, in the implication $(1 \implies 2)$, the decomposition is unique, i.e. *the context $\mathtt{C}'$, number of holes $n \geq 0$, and terms $s'_1, \dots, s'_n$ are the unique possible such objects.*

Proof. Remark that, in general, if $\mathtt{C}' \ltimes t$, where $\mathtt{C}'$ is a many-hole context and t is a term, then $\mathtt{C}'$ must be actually a 0-hole context. This can be easily checked by induction on $\mathtt{C}'$. We prove each direction separately:

- $(1 \implies 2)$ By induction on $\mathtt{C}$.
  1. **Empty,** $\mathtt{C} = \Box$**.** Then $t' \ltimes s$. Taking $\mathtt{C}' = \Box$ and $s'_1 := t'$, we have that $t' = \mathtt{C}'\langle s'_1\rangle$. Moreover, the decomposition is unique.
  2. **Abstraction,** $\mathtt{C} = \lambda x.\mathtt{C}_1$**.** Then t' must be of the form $\lambda^\ell x.u'$ where $u' \ltimes \mathtt{C}\langle s\rangle$. By *i.h.*, write u' as $u' = \mathtt{C}'_1\langle s'_1, \dots, s'_n\rangle$ where $\mathtt{C}'_1 \ltimes \mathtt{C}_1$ and $s'_i \ltimes s$ for all $1 \leq i \leq n$. Taking $\mathtt{C}' := \lambda^\ell x.\mathtt{C}'_1$ we conclude. Moreover, note that $\mathtt{C}'$ must start with $\lambda^\ell x.\Box$. By *i.h.* the decomposition is unique for s', so the decomposition is also unique for t'.
  3. **Left of an application,** $\mathtt{C} = \mathtt{C}_1\, u$**.** Then t' must be of the form $r'[u'_1, \dots, u'_m]$ with $r' \ltimes \mathtt{C}_1\langle s\rangle$ and $u'_i \ltimes u$ for all $1 \leq i \leq m$. By *i.h.*, write r' as $r' = \mathtt{C}'_1\langle s'_1, \dots, s'_n\rangle$ with $\mathtt{C}'_1 \ltimes \mathtt{C}_1$ and $s'_i \ltimes s$ for all $1 \leq i \leq n$. Taking $\mathtt{C}' := \mathtt{C}'_1\,[u'_1, \dots, u'_m]$ we conclude. Moreover, note that $\mathtt{C}'$ must start with $\Box[u'_1, \dots, u'_m]$ where, by the previous remark, all the u'_i must be 0-hole contexts. By *i.h.* the decomposition is unique for r', so the decomposition is also unique for t'.
  4. **Right of an application,** $\mathtt{C} = u\,\mathtt{C}_1$**.** Then t' must be of the form $u'\,[r'_1, \dots, r'_m]$ where $u' \ltimes u$ and $r'_i \ltimes \mathtt{C}_1\langle s\rangle$ for all $1 \leq i \leq m$. By *i.h.*, we can write each of the r'_i as $r'_i = \mathtt{C}'_{(1,i)}\langle s'_{(i,1)}, \dots, s'_{(i,k_i)}\rangle$ where $\mathtt{C}'_{(1,i)}$ is a context of k_i holes such that $\mathtt{C}'_{(1,i)} \ltimes \mathtt{C}_1$ and $s'_{(i,j)} \ltimes s$ for all $1 \leq i \leq m$ and all $1 \leq j \leq k_i$. Taking $\mathtt{C}' = u'[\mathtt{C}'_{(1,1)}, \dots, \mathtt{C}'_{(1,m)}]$ as a

context with $m = \sum_{i=1}^{n} k_i$ holes we conclude. Moreover, note that $\mathtt{C}'$ must start with $u'[\Box, \dots, \Box]$ where the application has exactly m arguments, and, by the previous remark, u' must be a 0-hole context. By *i.h.* the decomposition fo reach u'_i is unique, so the decomposition is also unique for t'.

- $(2 \implies 1)$ By induction on $\mathtt{C}$.
  1. **Empty,** $\mathtt{C} = \Box$**.** Then $\mathtt{C}' = \Box$ and $s'_1 = t'$, so we are done.
  2. **Abstraction,** $\mathtt{C} = \lambda x.\mathtt{C}_1$**.** Then $\mathtt{C}' = \lambda^{\ell} x.\mathtt{C}'_1$ where $\mathtt{C}'_1 \ltimes \mathtt{C}_1$. By *i.h.* $\mathtt{C}'_1\langle s'_1, \dots, s'_n\rangle \ltimes \mathtt{C}_1\langle s\rangle$, so $t' = \lambda^{\ell} x.\mathtt{C}'_1\langle s'_1, \dots, s'_n\rangle \ltimes \lambda x.\mathtt{C}_1\langle s\rangle$.
  3. **Left of an application,** $\mathtt{C} = \mathtt{C}_1\, u$**.** Then $\mathtt{C}' = \mathtt{C}'_1\,[u'_1, \dots, u'_m]$ where $\mathtt{C}'_1 \ltimes \mathtt{C}_1\langle s\rangle$ and $u'_i \ltimes u$ for all $1 \le i \le m$. Note that the u'_i are terms (*i.e.* they are 0-hole contexts), by the previous remark. Then by *i.h.*, $\mathtt{C}'_1\langle s'_1, \dots, s'_n\rangle \ltimes \mathtt{C}_1\langle s\rangle$, so $t' = \mathtt{C}'_1\langle s'_1, \dots, s'_n\rangle[u'_1, \dots, u'_m] \ltimes \mathtt{C}_1\langle s\rangle\, u$.
  4. **Right of an application,** $\mathtt{C} = u\,\mathtt{C}_1$**.** Then $\mathtt{C}' = u'[\mathtt{C}'_{(1,1)}, \dots, \mathtt{C}'_{(1,m)}]$ where $u' \ltimes u$ and $\mathtt{C}'_{(1,j)} \ltimes \mathtt{C}_1$ for all $1 \le j \le m$. Note that u' is a term (*i.e.* it is a 0-hole context), by the previous remark. Moreover, for each $1 \le j \le m$, the context $\mathtt{C}'_{(1,j)}$ is a context with $k_j \ge 0$ holes, in such a way that $\sum_{j=1}^{m} k_j = n$. Let us split the list $[s'_1, \dots, s'_n]$ in m lists, such that the j-th list has k_j elements, *i.e.* $[s'_1, \dots, s'_n] = [s'_{(1,1)}, \dots, s'_{(1,k_1)}, \dots, s'_{(m,1)}, \dots, s'_{(1,k_m)}]$. By *i.h.*, for each $1 \le j \le m$ we have that $\mathtt{C}'_{(1,j)}\langle s'_{(j,1)}, \dots, s'_{(j,k_j)}\rangle \ltimes \mathtt{C}_1\langle s\rangle$. So:

$$t' = u'[\mathtt{C}'_{(1,1)}\langle s'_{(1,1)}, \dots, s'_{(1,k_1)}\rangle, \dots, \mathtt{C}'_{(1,m)}\langle s'_{(m,1)}, \dots, s'_{(j,k_m)}\rangle] \ltimes u\,\mathtt{C}_1\langle s\rangle$$

  which concludes the proof.

□

Lemma A.15 (Contexts refined by terms may be filled)**.** *Suppose that $t' \ltimes \mathtt{C}$ where t' is a term (with no holes). Then $t' \ltimes \mathtt{C}\langle \mathtt{X}\rangle$ for any term or context $\mathtt{X}$.*

Proof. By induction on $\mathtt{C}$.

1. **Empty,** $\mathtt{C} = \Box$**.** This case is impossible, since it implies $t' = \Box$.
2. **Abstraction,** $\mathtt{C} = \lambda x.\mathtt{C}'$**.** Then $t' = \lambda^{\ell} x.t''$ where $t'' \ltimes \mathtt{C}'$. By *i.h.*, $t'' \ltimes \mathtt{C}'\langle \mathtt{X}\rangle$ so $t' = \lambda^{\ell} x.t'' \ltimes \lambda x.\mathtt{C}'\langle \mathtt{X}\rangle = \mathtt{C}\langle \mathtt{X}\rangle$.
3. **Left of an application,** $\mathtt{C} = \mathtt{C}'\, s$**.** Then $t' = t''\,[s'_1, \dots, s'_n]$ with $t'' \ltimes \mathtt{C}'$ and $s'_i \ltimes s$ for all $1 \le i \le n$. By *i.h.*, $t'' \ltimes \mathtt{C}'\langle \mathtt{X}\rangle$ so $t' = t''\,[s'_1, \dots, s'_n] \ltimes \mathtt{C}'\langle \mathtt{X}\rangle\, s = \mathtt{C}\langle \mathtt{X}\rangle$.
4. **Right of an application,** $\mathtt{C} = \mathtt{C}'\, s$**.** Then $t' = s'\,[t''_1, \dots, t''_n]$ with $s' \ltimes s$ and $t''_i \ltimes \mathtt{C}'$ for all $1 \le i \le n$. By *i.h.* $t''_i \ltimes \mathtt{C}'\langle \mathtt{X}\rangle$ for all $1 \le i \le n$, so $t' = s'\,[t''_1, \dots, t''_n] \ltimes s\,\mathtt{C}'\langle \mathtt{X}\rangle = \mathtt{C}\langle \mathtt{X}\rangle$.

□

A.12 Proof of Proposition 4.3 — Simulation

We want to prove that if $t, s \in \mathcal{T}^{\lambda}$ and $t' \in \mathcal{T}^{\#}$ be a distributive term such that $t' \ltimes t \to_{\beta} s$, then there is a distributive term $s' \in \mathcal{T}^{\#}$ such that $t' \to\!\!\!\!\to_{\#} s' \ltimes s$.

Proof. Let $t = \mathtt{C}\langle(\lambda x.p)q\rangle \to \mathtt{C}\langle p\{x := q\}\rangle = s$. We proceed by induction on $\mathtt{C}$.

1. **Empty context,** $\mathtt{C} = \Box$**.** Then $t = (\lambda x.p)q \to p\{x := q\} = s$. Then $t' = (\lambda^{\ell} x.p')[q'_1, \dots, q'_n]$, for some ℓ, $p' \ltimes p$ and $q'_i \ltimes q$.
   We can do the step $t' \to\!\!\!\!\to_{\#} p'\{x := [q'_1, \dots, q'_n]\}$. We choose s' to be the latter term, which by Lemma A.13 refines s.

2. **Under an abstraction,** $\mathtt{C} = \lambda y.\mathtt{C}'$**.** Then $t = \lambda y.\mathtt{C}'\langle(\lambda x.p)q\rangle \to \lambda y.\mathtt{C}'\langle p\{x := q\}\rangle = s$. If $t' \ltimes t$, then $t' = \lambda^{\ell} y.r$, with $r \ltimes \mathtt{C}'\langle(\lambda x.p)q\rangle$.
By inductive hypothesis, there is a term r' such that

$$\begin{array}{ccc} \mathtt{C}'\langle(\lambda x.p)q\rangle & \xrightarrow{\beta} & \mathtt{C}'\langle p\{x := q\}\rangle \\ \rotatebox{90}{$\trianglelefteq$} & & \rotatebox{90}{$\trianglelefteq$} \\ r & \overset{\#}{\twoheadrightarrow} & r' \end{array}$$

Then,

$$\begin{array}{ccc} \lambda y.\mathtt{C}'\langle(\lambda x.p)q\rangle & \xrightarrow{\beta} & \lambda y.\mathtt{C}'\langle p\{x := q\}\rangle \\ \rotatebox{90}{$\trianglelefteq$} & & \rotatebox{90}{$\trianglelefteq$} \\ \lambda^{\ell} y.r & \overset{\#}{\twoheadrightarrow} & \lambda^{\ell} y.r' \end{array}$$

3. **Left of an application,** $\mathtt{C} = \mathtt{C}'u$**.** Then $t = \mathtt{C}'\langle(\lambda x.p)q\rangle u \to \mathtt{C}'\langle p\{x := q\}\rangle u = s$. If $t' \ltimes t$, then $t' = r[u'_1, \ldots, u'_n]$, with $r \ltimes \mathtt{C}'\langle(\lambda x.p)q\rangle$ and $u'_i \ltimes u$.
By inductive hypothesis, there is a term r' such that

$$\begin{array}{ccc} \mathtt{C}'\langle(\lambda x.p)q\rangle & \xrightarrow{\beta} & \mathtt{C}'\langle p\{x := q\}\rangle \\ \rotatebox{90}{$\trianglelefteq$} & & \rotatebox{90}{$\trianglelefteq$} \\ r & \overset{\#}{\twoheadrightarrow} & r' \end{array}$$

Then,

$$\begin{array}{ccc} \mathtt{C}'\langle(\lambda x.p)q\rangle u & \xrightarrow{\beta} & \mathtt{C}'\langle p\{x := q\}\rangle u \\ \rotatebox{90}{$\trianglelefteq$} & & \rotatebox{90}{$\trianglelefteq$} \\ r[u'_1, \ldots, u'_n] & \overset{\#}{\twoheadrightarrow} & r'[u'_1, \ldots, u'_n] \end{array}$$

4. **Right of an application,** $\mathtt{C} = u\mathtt{C}'$**.** Then $t = u\mathtt{C}'\langle(\lambda x.p)q\rangle \to u\mathtt{C}'\langle p\{x := q\}\rangle = s$. If $t' \ltimes t$, then $t' = u'[u'_1, \ldots, u'_n]$, with $u' \ltimes u$ and $u'_i \ltimes \mathtt{C}'\langle(\lambda x.p)q\rangle$.
We can apply the inductive hypothesis to every u'_i, obtaining

$$\begin{array}{ccc} \mathtt{C}'\langle(\lambda x.p)q\rangle & \xrightarrow{\beta} & \mathtt{C}'\langle p\{x := q\}\rangle \\ \rotatebox{90}{$\trianglelefteq$} & & \rotatebox{90}{$\trianglelefteq$} \\ u'_i & \overset{\#}{\twoheadrightarrow} & u''_i \end{array}$$

Then,

$$\begin{array}{ccc} \mathtt{C}'\langle(\lambda x.p)q\rangle u & \xrightarrow{\beta} & \mathtt{C}'\langle p\{x := q\}\rangle u \\ \rotatebox{90}{$\trianglelefteq$} & & \rotatebox{90}{$\trianglelefteq$} \\ u'[u'_1, \ldots, u'_n] & \overset{\#}{\twoheadrightarrow} & u'[u''_1, \ldots, u''_n] \end{array}$$

Because

$$\begin{aligned} u'[u'_1,\dots,u'_{i-1},u'_i,u_{i+1},\dots,u'_n] &\twoheadrightarrow_{\#} u'[u''_1,\dots,u'_{i-1},u'_i,u'_{i+1},\dots,u'_n] \\ &\twoheadrightarrow_{\#} \dots \\ &\twoheadrightarrow_{\#} u'[u''_1,\dots,u''_{i-1},u'_i,u'_{i+1},\dots,u''_n] \\ &\twoheadrightarrow_{\#} u'[u''_1,\dots,u''_{i-1},u''_i,u'_{i+1},\dots,u''_n] \\ &\twoheadrightarrow_{\#} u'[u''_1,\dots,u''_{i-1},u''_i,u''_{i+1},\dots,u''_n] \\ &\twoheadrightarrow_{\#} \dots \\ &\twoheadrightarrow_{\#} u'[u''_1,\dots,u''_{i-1},u''_i,u''_{i+1},\dots,u''_n] \end{aligned}$$

□

A.13 Proof of Proposition 4.5 — Reverse simulation

We repeat the statement of reverse simulation in its entirety.

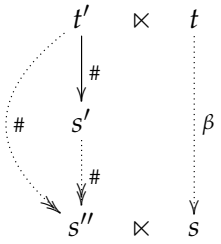
Let $t', s' \in \mathcal{T}^{\#}$ be distributive-terms and let $t \in \mathcal{T}^{\lambda}$ be a lambda-term such that:

$$t \rtimes t' \to_{\#} s'$$

then there is a distributive term $s'' \in \mathcal{T}^{\#}$ and a lambda-term $s \in \mathcal{T}^{\lambda}$ such that:

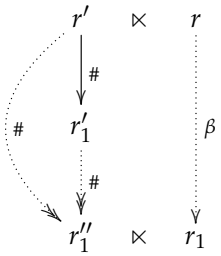
$$t \to_{\beta} s \rtimes s''$$

and the step $t' \to_{\#} s'$ is contained in the multistep $t' \twoheadrightarrow_{\#} s''$. Diagramatically:

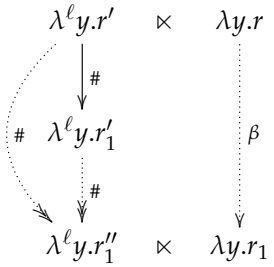


Proof. By induction on t'.

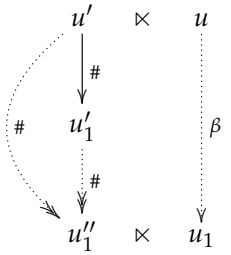
1. **Variable** $t' = x$**.** This case cannot happen because a variable cannot be reduced.
2. **Abstraction,** $t' = \lambda^{\ell} y.r'$**.** In this case $t = \lambda y.r$, where $r \rtimes r'$. Also, we have that $t' \to_{\#} s'$, so then necessarily $s' = \lambda^{\ell} y.r'_1$ where $r' \to_{\#} r'_1$.
   Then, by inductive hypothesis we have that



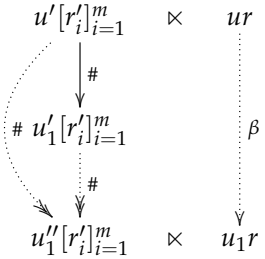
So adding a lambda at the beggining of every term in the diagram we get our desired result



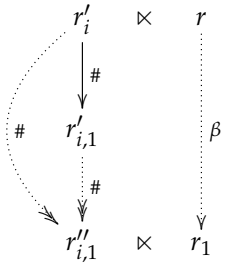
3. **Application,** $t' = u'[r'_1, \ldots, r'_m]$**.** In this case we have several subcases, depending on where the reduction step is done. It can be done inside u', inside a r'_i for some $i \in \{1, \ldots, m\}$, or in the head of the term (in the case that u' is a lambda). Let $R : t' \to_\# s'$ be the name of the step.
Also, $t = ur$, such that $u \rtimes u'$ and $r \rtimes r'_i$ for all $i \in \{1, \ldots, m\}$.
   3.1 R **is inside** u'**.** So then $u' \to_\# u'_1$, and by inductive hypothesis we have that



   So adding the arguments at the end of every term in the diagram we get our desired result



   3.2 R **is inside** r'_i **for some** $i \in \{1, \ldots, m\}$**.** We have that $r'_i \to_\# r'_{i,1}$, so by inductive hypothesis



Now, we would like to reduce the rest of the r'_j to something that refines r_1. Luckily, we can do that using simulation.
As for every $j \neq i$ we have that $r'_j \ltimes r$ and $r \to_\beta r_1$ then because of simulation we get the following diagram for every $j \neq i$.

$$\begin{array}{ccc} r & \xrightarrow{\beta} & r_1 \\ \rotatebox{90}{$\ltimes$} & & \rotatebox{90}{$\ltimes$} \\ r'_j & \overset{\#}{\dashrightarrow} & r'_{j,1} \end{array}$$

Using all those reductions we obtained, we can construct the following diagram, which proves what we wanted.

$$\begin{array}{ccc}
u'[r'_0, \ldots, r'_{i-1}, r'_i, r'_{i+1}, \ldots, r'_m] & \ltimes & ur \\
\downarrow \# & & \\
u'[r'_0, \ldots, r'_{i-1}, r'_{i,1}, r'_{i+1}, \ldots, r'_m] & & \\
\vdots \# & & \\
u'[r'_{0,1}, \ldots, r'_{i-1}, r'_{i,1}, r'_{i+1}, \ldots, r'_m] & & \\
\vdots \# & & \\
u'[r'_{0,1}, \ldots, r'_{i-1,1}, r'_{i,1}, r'_{i+1}, \ldots, r'_m] & & \\
\vdots \# & & \beta \\
u'[r'_{0,1}, \ldots, r'_{i-1,1}, r''_{i,1}, r'_{i+1}, \ldots, r'_m] & & \\
\vdots \# & & \\
u'[r'_{0,1}, \ldots, r'_{i-1,1}, r''_{i,1}, r'_{i+1,1}, \ldots, r'_m] & & \\
\vdots \# & & \\
u'[r'_{0,1}, \ldots, r'_{i-1,1}, r''_{i,1}, r'_{i+1,1}, \ldots, r'_m] & & \\
\vdots \# & & \\
u'[r'_{0,1}, \ldots, r'_{i-1,1}, r''_{i,1}, r'_{i+1,1}, \ldots, r'_{m,1}] & \ltimes & ur_1
\end{array}$$

3.3 R **is at the root.** In this case $u' = \lambda^\ell x.s$, and we have the following diagram.

$$\begin{array}{ccc} (\lambda^\ell x.s')[r'_1,\ldots,r'_n] & \ltimes & (\lambda x.s)r \\ \Big\downarrow{\scriptstyle \#} & & \Big\downarrow{\scriptstyle \beta} \\ s'\{x := [r'_1,\ldots,r'_n]\} & & s\{x := r\} \end{array}$$

But $s'\{x := [r'_1,\ldots,r'_n]\} \ltimes s\{x := r\}$ because of Lemma A.13, so we are done. □

A.14 Proof of Lemma 4.8 — Head normal forms have refinements

We had a term $t \in \mathcal{T}^\lambda$ that is in head normal form, *i.e.* $t = \lambda x_1.\ldots\lambda x_n.y\,t_1\,\ldots\,t_m$.
We wanted to prove that:

- that there exists $t' \in \mathcal{T}^\#$ such that $t' \ltimes t$,
- that for every application in t' the argument list is empty, and
- that if $x_i = y$ for any i, Γ will be empty, otherwise it will be a singleton of the form $\{y : [\tau]\}$.

Proof. We proceed by induction on the pair (n, m), that is, by induction on $\mathbb{N}^2$ with lexicographic order.

1. **Base** $(0,0)$**.** In this case $t = y$, so we can simply choose t' to be y^τ.
2. **Inductive step.** We want to see that the property holds for $(n,m) > (0,0)$ given that it holds for all $(n',m') < (n,m)$. Here there are two cases, either $n = 0$ or $n > 0$.
   2.1 $n = 0$. We have that $t = y\,t_1\,\ldots\,t_m$, $m \geq 1$. Let $s = y\,t_1\,\ldots\,t_{m-1}$. By inductive hypothesis, there exists a correct term s' that refines s.
   We can suppose, without loss of generality, that $s' = y^\tau\,\vec{t}_1\,\ldots\,\vec{t}_{m-1}$, where each element of $\vec{t}_i$ refines t_i. If s' wasn't like that, it simply would not refine s. If $m = 1$, then $s' = y$ and the type of y is τ_1.
   If $m > 1$, the inductive hypothesis tells us that all $\vec{t}_i$ are empty, so $s' = y[\,]_1\ldots[\,]_{m-1}$ (we number them to make it more clear which is which). Because of Unique typing, the type of y has to be $[\,]_1 \to \ldots \to [\,]_{m-1} \to \tau_1$, where τ_1 is the type of s'. Diagramatically,

$$\frac{\{y : [\,]_1 \to \ldots \to [\,]_{m-1} \to \tau_1\} \vdash y^{[\,]_1\to\ldots\to[\,]_{m-1}\to\tau_1}\,[\,]_1\,\ldots\,[\,]_{m-2} : [\,]_{m-1} \to \tau_1}{\{y : [\,]_1 \to \ldots \to [\,]_{m-1} \to \tau_1\} \vdash y^{[\,]_1\to\ldots\to[\,]_{m-1}\to\tau_1}\,[\,]_1\,\ldots\,[\,]_{m-1} : \tau_1}\,{\to_\mathrm{E}}$$

   Now, we can give a typing derivation for the following term, if we consider a derivation very similar to the last one but where τ_1 has been replaced by $[\,] \to \tau_2$.

$$\frac{\{y : [\,]_1 \to \ldots \to [\,]_{m-1} \to [\,] \to \tau_2\} \vdash y^{[\,]_1\to\ldots\to[\,]_{m-1}\to[\,]\to\tau_2}\,[\,]_1\,\ldots\,[\,]_{m-1} : [\,] \to \tau_2}{\{y : [\,]_1 \to \ldots \to [\,]_{m-1} \to [\,] \to \tau_2\} \vdash y^{[\,]_1\to\ldots\to[\,]_{m-1}\to[\,]\to\tau_2}\,[\,]_1\,\ldots\,[\,]_{m-1}\,[\,] : \tau_2}\,{\to_\mathrm{E}}$$

   Let $t' = y^{[\,]_1\to\ldots\to[\,]_{m-1}\to[\,]\to\tau_2}\,[\,]_1\,\ldots\,[\,]_{m-1}\,[\,]$.

Note that the derivation fo t' can be completed very easily in the same way that s' was done, and it's easy to check that t' is correct (contexts are sequential, there are no lambdas, and types are sequential, because the derivation is essentially the same that the one of s'). Moreover, t' refines t, so we are done.

2.2 $n > 0$. We have that $t = \lambda x_1 . \ldots \lambda x_n . y\, t_1 \ldots t_m$. Let $s = \lambda x_2 . \ldots \lambda x_n . y\, t_1 \ldots t_m$. By inductive hypothesis, there is a term $\Gamma \vdash s' : \tau$ that refines s. The same way as before, we know that $s' = \lambda^{\ell_2} x_2 . \ldots \lambda^{\ell_n} x_n . y\, [\,] \ldots [\,]$.
Let $t' = \lambda^{\ell} x_1 . s'$, where ℓ is fresh. If $x_1 = y$ and $x_1 \neq x_i$ for all $1 < i \leq n$, then ℓ is the external label of y, and in any other case it is a fresh label. Here to make the proof tidier we will divide in two new cases, depending on whether $x_1 = y$ and $x_1 \neq x_i$ for all $1 < i \leq n$, or not.

2.2.1 $x_1 = y$ **and** $x_1 \neq x_i$ **for all** $1 < i \leq n$**.** In this case, by inductive hypothesis Γ must be a singleton $\{y : \tau_1\}$.

$$\frac{\{y : \tau_1\} \vdash s' : \tau}{\varnothing \vdash \lambda^{\ell} x_1 . s' : [\tau_1] \xrightarrow{\ell} \tau} \to_{\mathrm{I}}$$

We have that $t' \ltimes t$, and that the derivation of t' has sequential types and contexts, because by inductive hypothesis s' does. Finally, t' has pairwise distinct labels in all lambdas because s' does and ℓ was chosen to be a fresh label.

2.2.2 $x_1 \neq y$ **or** $x_1 = x_i$ **for some** $1 < i \leq n$**.**

$$\frac{\{y : \tau_1\} \vdash s' : \tau}{\{y : \tau_1\} \vdash \lambda^{\ell} x_1 . s' : [\,] \xrightarrow{\ell} \tau} \to_{\mathrm{I}}$$

Like in the previous case, we have that $t' \ltimes t$, and that the derivation of t' has sequential types and contexts, because by inductive hypothesis s' does, and that t' has pairwise distinct labels in all lambdas because s' does and ℓ was chosen to be a fresh label.

□

A.15 Proof of Proposition 4.9 — Refinability characterizes head normalization

Before proving Proposition 4.9, we need a few auxiliary results.

Lemma A.16 ($\to_{\#}$-normal forms refine head normal forms)**.** *Let $t' \in \mathcal{T}^{\#}$ be a $\to_{\#}$-normal form and $t' \ltimes t$. Then t is a head normal form.*

Proof. Observe, by induction on t' that if $t' \in \mathcal{T}^{\#}$ is a $\to_{\#}$-normal form, it must be of the form $\lambda^{\ell_1} x_1 . \ldots \lambda^{\ell_n} x_n . y^{\tau} \vec{s}_1 \ldots \vec{s}_m$, as it cannot have a subterm of the form $(\lambda^{\ell'} z . u)\vec{r}$. Then

$$\lambda^{\ell_1} x_1 . \ldots \lambda^{\ell_n} x_n . y^{\tau} \vec{s}_1 \ldots \vec{s}_m \ltimes t$$

. So t is of the form $\lambda x_1 . \ldots \lambda x_n . y s_1 \ldots s_n$, that is, t is a head normal form. □

The following lemma is an adaptation of Subject Expansion in [BKV17].

Lemma A.17 (Subject Expansion). *If* $\Gamma \vdash \mathtt{C}\langle t\{x := \vec{s}\}\rangle : \tau$ *is derivable, then* $\Gamma \vdash \mathtt{C}\langle(\lambda^{\ell} x.t)\vec{s}\rangle : \tau$ *is derivable.*

Proof. The proof proceeds by induction on $\mathtt{C}$, and the base case by induction on t, similar to the proof that substitution preserves typing in the proof of Subject Reduction (Section A.4). □

Correctness is not necessarily preserved by $\to_{\#}$-expansion. We need a stronger invariant, **strong sequentiality**, that will be shown to be preserved by expansion under appropiate conditions:

Definition A.18 (Subterms and free subterms)**.** The set of **subterms** $\mathsf{sub}(t)$ of a term t is formally defined as follows:

$$\begin{array}{rcl} \mathsf{sub}(x^{\tau}) & \stackrel{\text{def}}{=} & \{x^{\tau}\} \\ \mathsf{sub}(\lambda^{\ell} x.t) & \stackrel{\text{def}}{=} & \{\lambda^{\ell} x.t\} \cup \mathsf{sub}(t) \\ \mathsf{sub}(t[s_i]_{i=1}^{n}) & \stackrel{\text{def}}{=} & \{t[s_i]_{i=1}^{n}\} \cup \mathsf{sub}(t) \cup \cup_{i=1}^{n} \mathsf{sub}(s_i) \end{array}$$

The set of **free subterms** $\mathsf{sub}^{\circ}(t)$ of a term t is defined similarly, except for the abstraction case, which requires that the subterm in question do not include occurrences of bound variables:

$$\begin{array}{rcl} \mathsf{sub}^{\circ}(x^{\tau}) & \stackrel{\text{def}}{=} & \{x^{\tau}\} \\ \mathsf{sub}^{\circ}(\lambda^{\ell} x.t) & \stackrel{\text{def}}{=} & \{\lambda^{\ell} x.t\} \cup \{u \in \mathsf{sub}^{\circ}(t) \mid x \notin \mathsf{fv}(u)\} \\ \mathsf{sub}^{\circ}(t[s_i]_{i=1}^{n}) & \stackrel{\text{def}}{=} & \{t[s_i]_{i=1}^{n}\} \cup \mathsf{sub}^{\circ}(t) \cup \cup_{i=1}^{n} \mathsf{sub}^{\circ}(s_i) \end{array}$$

Definition A.19 (Strong sequentiality)**.** A term t is **strongly sequential** if it is correct and, moreover, for every subterm $s \in \mathsf{sub}(t)$ and any two free subterms $s_1, s_2 \in \mathsf{sub}^{\circ}(s)$ lying at disjoint positions of s, the types of s_1 and s_2 have different external labels.

Example A.20. *The following examples illustrate the notion of strong sequentiality:*

1. *The term* $t = (\lambda^{1} x.y^{\alpha^2})[\,]$ *is strongly sequential. Note that* t *and* y^{α^2} *have the same type, namely* α^2*, but they do not occur at* disjoint *positions.*
2. *The term* $t = (\lambda^{1} x.x^{\alpha^2})[y^{\alpha^2}]$ *is strongly sequential. Note that* x^{α^2} *and* y^{α^2} *both have type* α^2*, but they are not simultaneously free subterms of the same subterm of* t*.*
3. *The term* $t = \lambda^{1} y.x^{[\alpha^2]\stackrel{3}{\to}[\alpha^2]\stackrel{4}{\to}\beta^5}[y^{\alpha^2}][z^{\alpha^2}]$ *is not strongly sequential, since* y^{α^2} *and* z^{α^2} *have the same type and they are both free subterms of* $x^{[\alpha^2]\stackrel{3}{\to}[\alpha^2]\stackrel{4}{\to}\beta^5}[y^{\alpha^2}][z^{\alpha^2}] \in \mathsf{sub}(t)$*.*

Lemma A.21 (Refinement of a substitution: decomposition)**.** *If* $u' \ltimes t\{x := s\}$ *and* u' *is strongly sequential, then* u' *is of the form* $t'\{x := [s'_i]_{i=1}^{n}\}$*. Moreover, given a fresh label* ℓ*, the term* $(\lambda^{\ell} x.t')[s'_i]_{i=1}^{n}$ *is strongly sequential,* $t' \ltimes t$ *and* $s'_i \ltimes s$ *for all* $i = 1..n$*.*

Proof. By induction on t.

1. **Variable (same),** $t = x$**.** Then $u' \ltimes s$. Let τ be the type of u'. Taking $t' := x^{\tau} \ltimes x$ we have that $(\lambda^{\ell} x.x^{\tau})[u']$ is strongly sequential. Regarding strong sequentiality, observe that x^{τ} and u' have the same type, but they are not simultaneously the free subterms of any subterm of $(\lambda^{\ell} x.x^{\tau})[u']$.
2. **Variable (different),** $t = y \neq x$**.** Then $u' \ltimes y$, so u' is of the form y^{τ}. Taking $t' := y^{\tau}$ we have that $(\lambda^{\ell} x.y^{\tau})[\,]$ is strongly sequential.

3. **Abstraction,** $t = \lambda y.r$**.** Then $u' \ltimes \lambda y.r\{x := s\}$ so u' is of the form $\lambda^{\ell'} y.u''$ where $u'' \ltimes r\{x := s\}$. By *i.h.*, u'' is of the form $r'\{x := [s'_i]_{i=1}^n\}$ where $(\lambda^{\ell} x.r')[s'_i]_{i=1}^n$ is strongly sequential, $r' \ltimes r$ and $s'_i \ltimes s$ for all $i = 1..n$. Taking $t' := \lambda^{\ell'} y.r'$, we have that $t' = \lambda^{\ell'} y.r' \ltimes \lambda y.r = t$. Moreover, the term $(\lambda^{\ell} x.t')[s_i]_{i=1}^n = (\lambda^{\ell} x.\lambda^{\ell'} y.r')[s_i]_{i=1}^n$ is strongly sequential. Typability is a consequence of Subject Expansion (Lemma A.17). The remaining properties are:
   3.1 *Uniquely labeled lambdas.* The multiset of labels decorating the lambdas of $(\lambda^{\ell} x.\lambda^{\ell'} y.r')[s_i]_{i=1}^n$ is given by $\Lambda((\lambda^{\ell} x.\lambda^{\ell'} y.r')[s_i]_{i=1}^n) = [\ell, \ell'] + \Lambda(r') +_{i=1}^n \Lambda(s_i)$. It suffices to check that this multiset has no repeats. The label ℓ is assumed to be fresh, so it occurs only once. By *i.h.*, $u'' = r'\{x := [s'_i]_{i=1}^n\}$, so using Lemma A.7 we have $\Lambda(u') = \Lambda(\lambda^{\ell'} y.u') = [\ell'] + \Lambda(u'') = [\ell'] + \Lambda(r'\{x := [s'_i]_{i=1}^n\}) = [\ell'] + \Lambda(r') +_{i=1}^n \Lambda(s'_i)$. Moreover, the term $u' = \lambda^{\ell'} y.u''$ is correct, so this multiset has no repeats.
   3.2 *Sequential contexts.* Let q be a subterm of $(\lambda^{\ell} x.\lambda^{\ell'} y.r')[s'_i]_{i=1}^n$. If q is a subterm of r' or a subterm of s'_i for some $i = 1..n$ we conclude by *i.h.* since $(\lambda^{\ell} x.r')[s'_i]_{i=1}^n$ is known to be correct. Moreover, if $\Gamma \oplus x : \mathcal{M} \oplus y : \mathcal{N}$ is the typing context for r', the typing contexts of $\lambda^{\ell'} y.r'$ and $\lambda^{\ell} x.\lambda^{\ell'} y.r'$ are respectively $\Gamma \oplus x : \mathcal{M}$ and Γ, which are also sequential. Finally, if Δ_i is the typing context for s'_i, for each $i = 1..n$, the typing context for $(\lambda^{\ell} x.r')[s'_i]_{i=1}^n$ is of the form $\Gamma +_{i=1}^n \Delta_i + y : \mathcal{N}$, and it is sequential by *i.h.*. Hence the typing context for the whole term is $\Gamma +_{i=1}^n \Delta_i$, and it is sequential.
   3.3 *Sequential types.* Let q be a subterm $(\lambda^{\ell} x.\lambda^{\ell'} y.r')[s'_i]_{i=1}^n$, and let $\mathcal{P} \xrightarrow{\ell''} \rho$ be a type that occurs in the typing context or the type of q. As in the previous case, we have that $\Gamma \oplus x : [\sigma_i]_{i=1}^n \oplus y : \mathcal{N} \vdash r' : \tau$ is derivable and $\Delta_i \vdash s'_i : \sigma_i$ is derivable for all $i = 1..n$. Moreover, they are correct by *i.h.*, so if q is a subterm of r' or a subterm of some s'_i, we are done. There are three cases left for q:
      3.3.1 **Case** $q = \lambda^{\ell'} y.r'$**.** The typing context is $\Gamma \oplus x : [\sigma_i]_{i=1}^n$ and the type $\mathcal{N} \xrightarrow{\ell'} \tau$.
      3.3.2 **Case** $q = \lambda^{\ell} x.\lambda^{\ell'} y.r'$**.** The typing context is Γ and the type $[\sigma_i]_{i=1}^n \xrightarrow{\ell} \mathcal{N} \xrightarrow{\ell'} \tau$.
      3.3.3 **Case** $q = (\lambda^{\ell} x.\lambda^{\ell'} y.r')[s_i]_{i=1}^n$**.** The typing context is Γ and the type $\mathcal{N} \xrightarrow{\ell'} \tau$.

      In all three cases, if $\mathcal{P} \xrightarrow{\ell''} \rho$ occurs in the typing context or the type of q, then $\mathcal{P}$ can be shown to be sequential using the *i.h.*.
   3.4 *Strong sequentiality.* Let $q \in \mathsf{sub}((\lambda^{\ell} x.\lambda^{\ell'} y.r')[s_i]_{i=1}^n)$ be a subterm, and let $q_1, q_2 \in \mathsf{sub}^{\circ}(q)$ be free subterms lying at disjoint positions of q. We argue that the types of q_1 and q_2 have different external labels. Consider the following five possibilities for q_1:
      3.4.1 **Case** $q_1 = (\lambda^{\ell} x.\lambda^{\ell'} y.r')[s_i]_{i=1}^n$**.** Impossible since q_2 must be at a disjoint position.
      3.4.2 **Case** $q_1 = \lambda^{\ell} x.\lambda^{\ell'} y.r'$**.** Then the external label of the type of q_1 is the label ℓ, which is fresh, so it cannot coincide with the type of any other subterm.
      3.4.3 **Case** $q_1 = \lambda^{\ell'} y.r'$**.** Then q_2 must be a subterm of s_i for some $i = 1..n$. Note that q must be the whole term, and $n > 0$, so there is at least one free occurrence of x in $\lambda^{\ell'} y.r'$. This means that $q_1 \notin \mathsf{sub}^{\circ}(q)$, so this case is impossible.
      3.4.4 **Case** q_1 **is a subterm of** r'**.** If q_2 is also a subterm of r', we conclude since by *i.h.* $(\lambda^{\ell} x.r')[s'_i]_{i=1}^n$ is strongly sequential. Otherwise, q_2 is a subterm of s_i for some $i = 1..n$, and we also conclude by *i.h.*.
      3.4.5 **Case** q_1 **is a subterm of** s_i **for some** $i = 1..n$**.** If q_2 is a subterm of s_j for some $j = 1..n$, we conclude since by *i.h.* $(\lambda^{\ell} x.r')[s'_i]_{i=1}^n$ is strongly sequential. If q_2 is any other subterm, note that the symmetric case has already been considered in

one of the previous cases.

4. **Application,** $t = rp$**.** Then $u' \ltimes (rp)\{x := s\}$, so it is of the form $u'_0[u'_j]_{j=1}^m$ where $u'_0 \ltimes r\{x := s\}$ and $u'_j \ltimes p\{x := s\}$ for all $j = 1..m$. By *i.h.* we have that u'_0 is of the form $r'\{x := \vec{s}_0\}$ and for all $j = 1..m$ the term u'_j is of the form $p'_j\{x := \vec{s}_j\}$, where $r' \ltimes r$ and $p'_j \ltimes p$ for all $j = 1..m$. Moreover, the length of the list $\vec{s}_j$ is k_j and $\vec{s}_j = [s_i^{(j)}]_{i=1}^{k_j}$ for all $j = 0..m$, and we have that $s_i^{(j)} \ltimes s$ for all $j = 0..m, i = 1..k_j$. By *i.h.* we also know that $(\lambda^\ell x.r')\vec{s}_0$ is strongly sequential and $(\lambda^\ell x.p'_j)\vec{s}_j$ is strongly sequential for all $j = 1..m$.
Let $t' := r'[p'_j]_{j=1}^m$, let $n := \sum_{j=0}^m k_j$, and let $[s'_1, \ldots, s'_n] := \sum_{j=0}^m \vec{s}_j$. Note that $t' = r'[p'_j]_{j=1}^m \ltimes rp = t$ and $s'_i \ltimes s$ for all $i = 1..n$.
Moreover, we have to check that $u' = t'\{x := [s'_1, \ldots, s'_n]\}$. To prove this, note that $t'\{x := [s'_1, \ldots, s'_n]\} = (r'[p'_j]_{j=1}^m)\{x := [s'_1, \ldots, s'_n]\}$. Suppose that the multiset $\mathtt{T}([s'_1, \ldots, s'_n])$ were sequential. Then the list of terms $[s'_1, \ldots, s'_n]$ would be partitioned as $(\vec{u}_0, \ldots, \vec{u}_m)$ where $\vec{u}_j$ is a permutation of $\vec{s}_j$ for all $j = 0..m$, and we would have indeed:

$$\begin{array}{lcll} t'\{x := [s'_1, \ldots, s'_n]\} & = & (r'[p'_j]_{j=1}^m)\{x := [s'_1, \ldots, s'_n]\} & \\ & = & r'\{x := \vec{u}_0\}[p'_j\{x := \vec{u}_j\}]_{j=1}^m) & \text{by } \mathtt{T}([s'_1, \ldots, s'_n]) \text{ sequential} \\ & = & r'\{x := \vec{s}_0\}[p'_j\{x := \vec{s}_j\}]_{j=1}^m) & \\ & = & r'\{x := \vec{s}_0\}[p'_j\{x := \vec{s}_j\}]_{j=1}^m) & \text{by Lemma A.3} \\ & = & u'_0[u'_j]_{j=1}^m = u' & \text{by } \textit{i.h.} \end{array}$$

To see that $\mathtt{T}([s'_1, \ldots, s'_n])$ is sequential, note that for every $i \neq j$, the terms s'_i and s'_j are free subterms of u' and they lie at disjoint positions of u'. Since u' is strongly sequential, the types of s'_i and s'_j have different external labels. Hence $\mathtt{T}([s'_1, \ldots, s'_n])$ is sequential.

To conclude, we are left to check that $(\lambda^\ell x.t')[s'_1, \ldots, s'_n]$ is strongly sequential:

4.1 *Uniquely labeled lambdas.* The multiset of labels decorating the lambdas of $(\lambda^\ell x.t')[s'_1, \ldots, s'_n]$ is given by $\Lambda((\lambda^\ell x.t')[s'_1, \ldots, s'_n]) = [\ell] + \Lambda(t') +_{i=1}^n \Lambda(s'_i)$. It suffices to check that this multiset has no repeats. The label ℓ is assumed to be fresh, so it occurs only once. We have already argued that $u' = t'\{x := [s'_1, \ldots, s'_n]\}$, so using Lemma A.7 we have $\Lambda(u') = \Lambda(t') +_{i=1}^n \Lambda(s'_i)$. Moreover u' is correct, so this multiset has no repeats.

4.2 *Sequential contexts.* Suppose that $\Gamma_0 \oplus x : \mathcal{M}_0 \vdash r' : [\rho_j]_{j=1}^m \xrightarrow{\ell'} \tau$ is derivable, $\Gamma_j \oplus x : \mathcal{M}_j \vdash p'_j : \rho_j$ is derivable for all $j = 1..m$, and $\Delta_i \vdash s'_i : \sigma_i$ is derivable for all $i = 1..n$. Note that $\sum_{j=0}^m \mathcal{M}_j = [\sigma_i]_{i=1}^n$.
Let q be a subterm of $(\lambda^\ell x.r'[p'_j]_{j=1}^m)[s'_1, \ldots, s'_n]$. Consider four cases for q:

4.2.1 **Case** $q = (\lambda^\ell x.r'[p'_j]_{j=1}^m)[s'_1, \ldots, s'_n]$**.** The typing context is $\sum_{j=0}^m \Gamma_j + \sum_{i=1}^n \Delta_i$. By Subject Expansion (Lemma A.17) the typing context of $(r'[p'_j]_{j=1}^m)\{x := [s'_1, \ldots, s'_n]\} = u'$ is also $\sum_{j=0}^m \Gamma_j + \sum_{i=1}^n \Delta_i$ and u' is correct by hypothesis. So $\sum_{j=0}^m \Gamma_j + \sum_{i=1}^n \Delta_i$ is sequential.

4.2.2 **Case** $q = \lambda^\ell x.r'[p'_j]_{j=1}^m$**.** The typing context is $\sum_{j=0}^m \Gamma_j$, which is sequential because $\sum_{j=0}^m \Gamma_j + \sum_{i=1}^n \Delta_i$ is sequential.

4.2.3 **Case** $q = r'[p'_j]_{j=1}^m$**.** The typing context is $\sum_{j=0}^m \Gamma_j \oplus x : [\sigma_i]_{i=1}^n$, which is sequential because $\sum_{j=0}^m \Gamma_j$ is sequential and, moreover, $[\sigma_i]_{i=1}^n = \mathtt{T}([\sigma'_1, \ldots, \sigma'_n])$ which we have already shown to be sequential.

4.2.4 **Otherwise.** Then q is a subterm of r', a subterm of p'_j for some $j = 1..m$, or a

subterm of some s'_i for some $i = 1..n$. Then we conclude since by *i.h.* $(\lambda^{\ell} x.r')\vec{s}_0$ and all the $(\lambda^{\ell} x.p'_j)\vec{s}_j$ are strongly sequential.

4.3 *Sequential types.* Let q be a subterm of $(\lambda^{\ell} x.r'[p'_j]_{j=1}^{m})[s'_1, \dots, s'_n]$. We claim that if $\mathcal{N} \xrightarrow{\ell''} \phi$ occurs in the context or in the type of q, then $\mathcal{N}$ is sequential. The proof is similar as for subcase .

4.4 *Strong sequentiality.* Let $q \in \mathsf{sub}((\lambda^{\ell} x.r'[p'_j]_{j=1}^{m})[s'_1, \dots, s'_n])$ be a subterm, and let $q_1, q_2 \in \mathsf{sub}^{\circ}(q)$ be free subterms lying at disjoint positions of q. We claim that the types of q_1 and q_2 have different external labels. The proof is similar as for subcase .

□

Lemma A.22 (Backwards Simulation)**.** *Let $t, s \in \mathcal{T}^{\lambda}$ be λ-terms and let $s' \in \mathcal{T}^{\#}$ be a strongly sequential term such that $t \to_{\beta} s$ and $s' \ltimes s$. Then there exists a strongly sequential term $t' \in \mathcal{T}^{\#}$ such that:*

$$\begin{array}{ccc} t & \xrightarrow{\beta} & s \\ \rotatebox{90}{$\ltimes$} & & \rotatebox{90}{$\ltimes$} \\ t' & \overset{\#}{\twoheadrightarrow} & s' \end{array}$$

Proof. Let $t = \mathtt{C}\langle(\lambda x.u)r\rangle \to_{\beta} \mathtt{C}\langle u\{x := r\}\rangle = s$. The proof proceeds by induction on $\mathtt{C}$.

1. **Empty, $\mathtt{C} = \Box$.** By Lemma A.21 we have that s' is of the form $u'\{x := [r'_1, \dots, r'_n]\}$ where $u' \ltimes u$ and $r'_i \ltimes r$ for all $i = 1..n$. Moreover, taking ℓ to be a fresh label, $(\lambda^{\ell} x.u')[r'_1, \dots, r'_n]$ is strongly sequential and $(\lambda^{\ell} x.u')[r'_1, \dots, r'_n] \ltimes (\lambda x.u)r$.
2. **Under an abstraction, $\mathtt{C} = \lambda x.\mathtt{C}'$.** Straightforward by *i.h.*.
3. **Left of an application, $\mathtt{C} = \mathtt{C}'\, u$** Straightforward by *i.h.*.
4. **Right of an application, $\mathtt{C} = u\, \mathtt{C}'$** Then $t = u\, r \to_{\beta} u\, p = s$ where $r \to_{\beta} p$ and $s' \ltimes s$. Then s' is of the form $u'[p'_1, \dots, p'_n]$ where $p'_i \ltimes p$ for all $i = 1..n$. By *i.h.*, for all $i = 1..n$ we have that there exist $r'_1, \dots, r'_n$ such that:

$$\begin{array}{ccc} r & \xrightarrow{\beta} & p \\ \rotatebox{90}{$\ltimes$} & & \rotatebox{90}{$\ltimes$} \\ r'_i & \overset{\#}{\twoheadrightarrow} & p'_i \end{array} \qquad \text{So we have:} \qquad \begin{array}{ccc} u\, r & \xrightarrow{\beta} & u\, p \\ \rotatebox{90}{$\ltimes$} & & \rotatebox{90}{$\ltimes$} \\ u'[r'_i]_{i=1}^{n} & \overset{\#}{\twoheadrightarrow} & u'[p'_i]_{i=1}^{n} \end{array}$$

Moreover, $u'[r'_i]_{i=1}^{n}$ is strongly sequential, which can be concluded from the facts that $u'[p'_i]_{i=1}^{n}$ is strongly sequential by hypothesis, r'_i is strongly sequential for all $i = 1..n$ by *i.h.*, and r'_i and p'_i have the same types by Subject Expansion (Lemma A.17).

□

To prove Proposition 4.9 recall that it states that the following are equivalent:

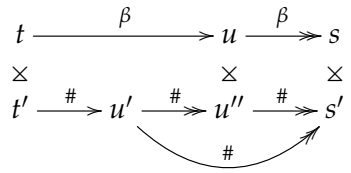
1. There exists $t' \in \mathcal{T}^{\#}$ such that $t' \ltimes t$.
2. There exists $t' \in \mathcal{T}^{\#}$ such that $t' \ltimes t$ and $t' \to_{\#}{}^{*} \lambda^{\ell_1} x_1. \dots \lambda^{\ell_n} x_n.y^{\tau}[] \dots []$.
3. There exists a head normal form s such that $t \to_{\beta}{}^{*} s$.

Let us prove the chain of implications $1 \implies 3 \implies 2 \implies 1$:

- $(1 \implies 3)$ Let $t' \ltimes t$. By Strong Normalization (Proposition 2.21), reduce $t' \to_{\#}{}^{*} s'$ to normal form. We claim that there exists a term s such that $t \to_{\beta}{}^{*} s$ and $s' \ltimes s$. Observe that, since $\lambda^{\#}$ is strongly normalizing (Proposition 2.21) and finitely branching, König's lemma ensures that there is a bound for the length of $\to_{\#}$-derivations going out from a term $t' \in \mathcal{T}^{\#}$. (Alternatively, according to the

proof of Strong Normalization in Proposition 2.21, the bound may be explicitly taken to be the number of lambdas in t'). Call this bound the **weight** of t'.

We proceed by induction on the weight of t'. If the derivation is empty, we are done by taking $s := t$. If the derivation is non-empty, it is of the form $t' \to_{\#} u' \to_{\#}^* s'$. By Simulation (Proposition 4.3) there exist terms u and u'' such that $u' \to_{\#}^* u'' \ltimes u$ and $t \to_\beta u$. Since the $\lambda^{\#}$-calculus is confluent (Proposition 2.27) and s' is a normal form, we have that $u'' \twoheadrightarrow_{\#} s'$. Note that the weight of t' is strictly larger than the weight of u'', so by *i.h.* there exists s such that $u \to_\beta^* s$ and $s' \ltimes s$:



Finally, since s' is a $\to_{\#}$-normal form and $s' \ltimes s$, Lemma A.16 ensures that s is a head normal form, as required.

- $(2 \implies 1)$ Obvious.
- $(3 \implies 2)$ Let $t \to_\beta^* s$ be a derivation to head normal form. We claim that there exists $t' \in \mathcal{T}^{\#}$ such that t' is strongly sequential, and the normal form of t' is of the form $\lambda^{\ell_1} x_1 . \ldots \lambda^{\ell_n} x_n . y^\tau [] \ldots []$. By induction on the length of the derivation $t \to_\beta^* s$. If the derivation is empty, $t = s$ is a head normal form and we conclude by Lemma 4.8, observing that the constructed term $t' \ltimes t$ is strongly sequential. If the derivation is non-empty, conclude using the *i.h.* and Backwards Simulation (Lemma A.22).

A.16 Proof of Lemma 4.15 — Basic cube lemma for simulation residuals

We are to prove that if $R : t \to s$ and $S : t \to u$ are coinitial steps, and $t' \in \mathcal{T}^{\#}$ is a correct term such that $t' \ltimes t$. Then the following equality between sets of coinitial steps holds:

$$(R/t')/(S/t') = (R/S)/(t'/S)$$

Recall that there are four notions of residual involved:

$$(R \,/^{(1)}\, t') \,/^{(2)}\, (S \,/^{(1)}\, t') = (R \,/^{(3)}\, S) \,/^{(1)}\, (t' \,/^{(4)}\, S)$$

1. Set of simulation residuals of a β-step relative to a correct term.
2. Set of residuals of a #-step after a #-step.
3. Set of residuals of a β-step after a β-step.
4. Simulation residual of a correct term after a β-step.

An auxiliary lemma is proven afterwards.

Proof. If $R = S$ then it is easy to see that the proposition holds, so we can assume that $R \neq S$. Also, note that it is enough to see that $\mathsf{names}((R/t')/(S/t')) = \mathsf{names}(((R/S)/(t'/S))$, as we will do that in some cases. We proceed by induction on t.

1. **Variable** $t = x$**.** This case is trivial (there are no steps with a variable as a source).
2. **Abstraction,** $t = \lambda x.u$**.** In this case we have $R : \lambda x.u \to \lambda x.u_1$ and $S : \lambda x.u \to \lambda x.u_2$. It also must be the case that:
   - $t' = \lambda^\ell x.u'$,
   - $R/t' : \lambda^\ell x.u' \to \lambda^\ell x.u'_1$, and
   - $S/t' : \lambda^\ell x.u' \to \lambda^\ell x.u'_2$.

   Where $u' \ltimes u$, $u'_1 \ltimes u_1$, and $u'_2 \ltimes u_2$. Note that using all that we can apply the inductive hypothesis on u (taking R and S to be the same steps but restricted to u). The inductive hypothesis yields that $(R/u')/(S/u') = (R/S)/(u'/S)$, which we can trivially extend to $(R/t')/(S/t') = (R/S)/(t'/S)$, because there aren't any other subterms in t' that may contain redexes other than u'.
3. **Application,** $t = pq$**.** We will look at several cases, depending on where R and S are located.

   3.1 R **is at the root.** So we have that $t = (\lambda x.r)q$, and $R : (\lambda x.r)q \to r\{x := q\}$.

   3.1.1 S **is in** r**.** In this case $r = \mathtt{C}\langle(\lambda y.u)v\rangle$.

$$\begin{array}{ccc} (\lambda x.\mathtt{C}\langle(\lambda y.u)v\rangle)q & \xrightarrow{\quad R \quad} & \mathtt{C}^\circ\langle(\lambda y.u^\circ)v^\circ\rangle \\ & \searrow^{S} & \\ \asymp & & (\lambda x.\mathtt{C}\langle u\{y := v\}\rangle)q \end{array}$$

$$\begin{array}{ccc} (\lambda^\ell x.\mathtt{C}'\langle(\lambda^{\ell_1} y.u_1)v_1, \ldots, (\lambda^{\ell_n} y.u_n)v_n\rangle)\vec{q} & \xrightarrow{R/t'} & \mathtt{C}'^\circ\langle(\lambda^{\ell_1} y.u_1^\circ)v_1^\circ, \ldots, (\lambda^{\ell_n} y.u_n^\circ)v_n^\circ\rangle) \\ & \searrow^{S/t'} & \\ & & (\lambda^\ell x.\mathtt{C}'\langle u_1\{y := v_1\}, \ldots, u_1\{y := v_1\}\rangle)\vec{q} \end{array}$$

   Note that $(R/t')/(S/t')$ is a set comprised of only one element, and that element is the step that reduces the lambda labeled with ℓ. Note that R/S also happens to have only one element, $(\lambda x.\mathtt{C}\langle u\{y := v\}\rangle)q \to \mathtt{C}^\circ\langle u^\circ\{y := v^\circ\}\rangle$, and is easy to see that the simulation of that step onto $(\lambda^\ell x.\mathtt{C}'\langle u_1\{y := v_1\}, \ldots, u_1\{y := v_1\}\rangle)\vec{q}$ yields the step that reduces the ℓ-lambda, *i.e.*, the desired step.

   3.1.2 S **is in** q**.** In this case $q = \mathtt{C}\langle(\lambda y.u)v\rangle$, so $t = (\lambda x.r)\mathtt{C}\langle(\lambda y.u)v\rangle$. Also, by Lemma A.14, t' must be of the form $(\lambda^\ell x.r')[\mathtt{C}_i\langle(\lambda^{\ell_{i,1}} y.u_{i,1})v_{i,1}, \ldots, (\lambda^{\ell_{i,m_i}} y.u_{i,m_i})v_{i,m_i}, \rangle]_{i=1}^n$.

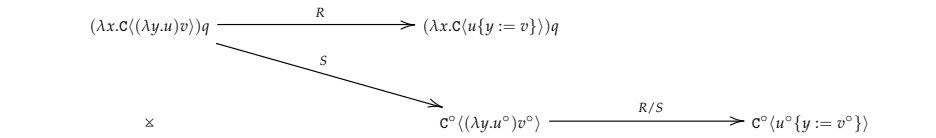
$$\begin{array}{ccc} (\lambda x.r)\mathtt{C}\langle(\lambda y.u)v\rangle & \xrightarrow{\quad R \quad} & r\{x := \mathtt{C}\langle(\lambda y.u)v\rangle\} \\ & \searrow^{S} & \\ & & (\lambda x.r)\mathtt{C}\langle u\{y := v\}\rangle \end{array}$$

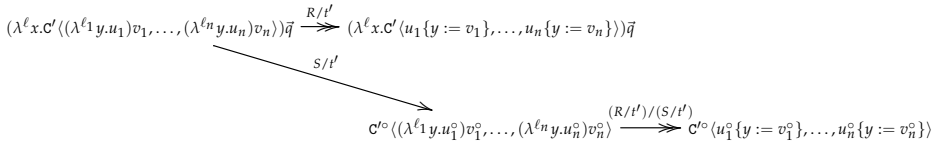
   Note that $(R/t')/(S/t')$, as before, is a set comprised of only one element and that element is the step that reduces the lambda labeled with ℓ, and has source

$t'/S = (\lambda^\ell x.r')[\mathtt{C}_i\langle u_{i,1}\{y := v_{i,1}\}, \ldots, u_{i,m_i}\{y := v_{i,m_i}\}\rangle]_{i=1}^n$. Note that R/S also happens to have only one element, $((\lambda x.r)\mathtt{C}\langle u\{y := v\}\rangle \to r\{x := \mathtt{C}\langle u\{y := v\}\rangle\})$ and is easy to see that the simulation of that step onto t'/S yields the step that reduces the ℓ-lambda, *i.e.*, the desired step.

3.2 R **is in** p**.** We separate in cases depending on where S is located.

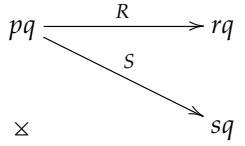
3.2.1 S **is at the root.** Here we have that $p = \lambda x.\mathtt{C}\langle(\lambda y.u)v\rangle$, hence the following diagram.

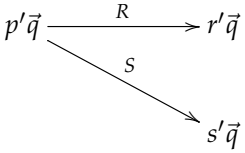




As seen in the diagram, $\mathsf{names}((R/t')/(S/t')) = \{\ell_1, \ldots, \ell_n\}$. But the lambda reduced by R/S is refined by each $\lambda^{\ell_i} y.u_i$, then $\mathsf{names}((R/S)/(t'/S)) = \{\ell_1, \ldots, \ell_n\}$, proving that the two sets are the same.

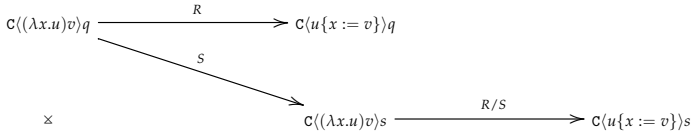
3.2.2 S **is in** p**.** This case is summarized by the following diagram.

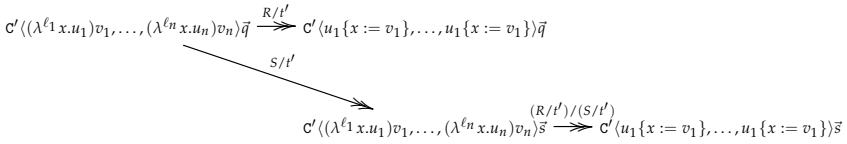




By inductive hypothesis, restricting R and S to p, we obtain $(R/p')/(S/p') = (R/S)/(p'/S)$, which we can extend to t', yielding the desired result.

3.2.3 S **is in** q**.** Again, we can summarize this case with a diagram.

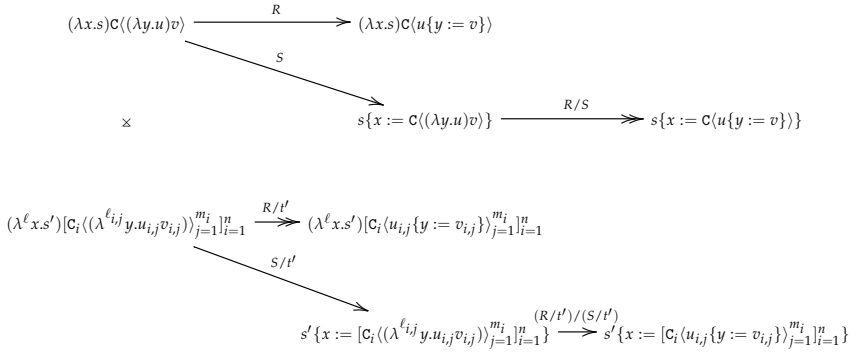




We know that $\frac{(\lambda^{\ell_i} x.u_i)v_i}{R/S} = u_i\{x := v_i\}$. Then by Lemma A.23, $\mathtt{C}'\langle(\lambda^{\ell_1} x.u_1)v_1, \ldots, (\lambda^{\ell_n} x.u_n)v_n\rangle/(R/S)$ equals $\mathtt{C}'\langle u_1\{x := v_1\}, \ldots, u_1\{x := v_1\}\rangle$.
In other words, $\mathsf{tgt}((R/S)/(t'/S)) = \mathsf{tgt}((R/t')/(S/t'))$, which is enough because their sources are the same by hypothesis.

3.3 R **is in** q**.** As before, we separate in cases depending on where S is located.

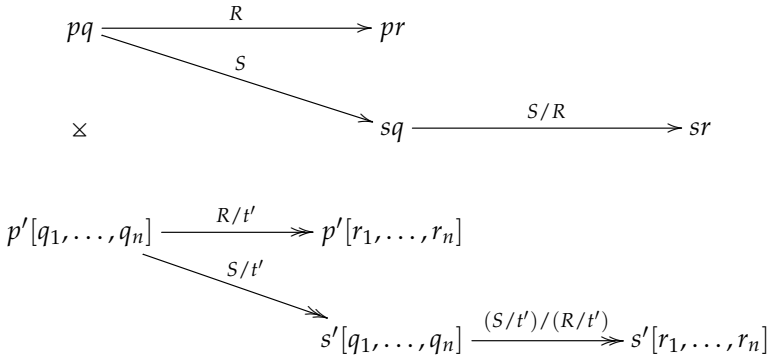
3.3.1 S **is at the root.** Then $p = \lambda x.s$ and $t = (\lambda x.s)\mathtt{C}\langle(\lambda x.u)v\rangle$.



First, note that R/S has as many elements as xs in s, in other words, S may erase or multiply R. Some of those xs are refined in t' (the number is exactly n, *i.e.*, the cardinality of the argument of the application).
Also, for each step in R/S, using Lemma A.23, we know that its projection onto t'/S yields the steps with name $\{e_{i,1}, \ldots, e_{i,m_i}\}$. Hence,

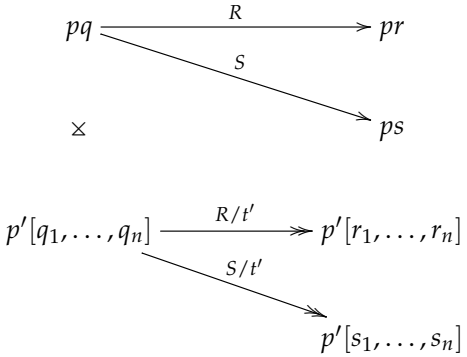
$$\mathsf{names}((R/S)/(t'/S)) = \cup_{i=1}^{n}\{e_{i,1}, \ldots, e_{i,m_i}\} = \mathsf{names}((R/t')/(S/t'))$$

3.3.2 S **is in** p**.** This case can be summarized using a diagram.



And can be solved in a similar manner to 3.2.3., using the same reasoning for each element of the argument of the application.

3.3.3 S **is in** q**.** The situation is as follows.



By inductive hypothesis, we know that for each $i \in \{1, \ldots, n\}$, $(R/S)/(q_i/S) = (R/q_i)/(S/q_i)$.
The diagram also shows that $\mathsf{names}((R/t')/(S/t')) = \bigcup_{i=1}^{n} \mathsf{names}((R/q_i)/(S/q_i))$, because we have to execute those steps and the calculus has no erasure or duplication.
Finally, by Lemma A.23, $\mathsf{names}((R/S)/(t'/S)) = \bigcup_{i=1}^{n} \mathsf{names}((R/S)/(q_i/S))$, so we can join all previous equalities, and obtain $\mathsf{names}((R/S)/(t'/S)) = \mathsf{names}((R/t')/(S/t'))$.

□

Lemma A.23 (Simulation of contexts)**.** *Let* $\mathtt{C}'$ *be an* n*-hole context in* $\mathcal{T}^{\#}$ *and* $\mathtt{C}$ *be a context in* $\mathcal{T}^{\lambda}$*. Also let* $t_1, \ldots, t_n \in \mathcal{T}^{\#}$ *and* $t \in \mathcal{T}^{\lambda}$ *such that* $t_i \ltimes t$ *for each* $i \in \{1, \ldots, n\}$*,* $\mathtt{C}' \ltimes \mathtt{C}$*, and* $\mathtt{C}'\langle t_1, \ldots, t_n\rangle \ltimes \mathtt{C}\langle t\rangle$*. Also, let* $R : \mathtt{C}\langle t\rangle \to \mathtt{C}\langle s\rangle$*.*

Then, $\mathtt{C}'\langle t_1, \ldots, t_n\rangle/R = \mathtt{C}'\langle t_1/R, \ldots, t_n/R\rangle$*, and* $\mathsf{names}(R/\mathtt{C}'\langle t_1, \ldots, t_n\rangle) = \bigcup_{i=1}^{n} \mathsf{names}(R/t_i)$*.*

Proof. By induction on $\mathtt{C}$.

1. $\mathtt{C} = \Box$**.** There is nothing to check here, as the result is trivially true: $t_1/R = t_1/R$ and $\mathsf{names}(R/t_1) = \bigcup_{i=1}^{1} \mathsf{names}(R/t_i)$.
2. $\mathtt{C} = \lambda x.\mathtt{C}''$**.** This case is straighforward using the inductive hypothesis.
3. $\mathtt{C} = r\mathtt{C}''$**.** Using Lemma A.14 this case reduces to the following diagram.

$$\begin{array}{ccc} r\mathtt{C}''\langle t\rangle & \xrightarrow{\quad R \quad} & r\mathtt{C}''\langle s\rangle \\ \ltimes & & \\ r[\mathtt{C}_1\langle t_{1,1}, \ldots, t_{1,m_1}\rangle, \ldots, \mathtt{C}_n\langle t_{n,1}, \ldots, t_{n,m_n}\rangle] & & \end{array}$$

By inductive hypothesis, $\mathtt{C}_i\langle t_{i,1}, \ldots, t_{i,m_i}\rangle/R = \mathtt{C}_i\langle t_{i,1}/R, \ldots, t_{i,m_i}/R\rangle$ and $\mathsf{names}(R/\mathtt{C}_i\langle t_{i,1}, \ldots, t_{i,m_i}\rangle) = \bigcup_{i=1}^{n} \mathsf{names}(R/t_i)$.
Finally, by the construction of the simulation,

$$r[\mathtt{C}_1\langle t_{1,1}, \ldots, t_{1,m_1}\rangle, \ldots, \mathtt{C}_n\langle t_{n,1}, \ldots, t_{n,m_n}\rangle]/R = r[\mathtt{C}_1\langle t_{1,1}, \ldots, t_{1,m_1}\rangle/R, \ldots, \mathtt{C}_n\langle t_{n,1}, \ldots, t_{n,m_n}\rangle/R]$$

and

$$\mathsf{names}(R/(r[\mathtt{C}_1\langle t_{1,1}, \ldots, t_{1,m_1}\rangle, \ldots, \mathtt{C}_n\langle t_{n,1}, \ldots, t_{n,m_n}\rangle])) = \bigcup_{i=1}^{n} \mathsf{names}(R/\mathtt{C}_1\langle t_{i,1}, \ldots, t_{i,m_i}\rangle)$$

which finishes our proof.

4. $\mathtt{C} = \mathtt{C}''r$. This case is straightforward using the inductive hypothesis.

□

A.17 Proof of Proposition 4.17 — Compatibility

We are to prove that the operation of taking simulation residuals on two derivations of the lambda calculus is compatible with permutation equivalence. With symbols, if $R \equiv S$ then $R/t' \equiv S/t'$.

We first prove the result for complete developments and then for arbitrary derivations.

Lemma A.24 (Compatibility for developments). *Let $\mathcal{M}$ be a set of coinitial steps, and let ρ and σ be complete developments of $\mathcal{M}$, and let $t' \in \mathcal{T}^{\#}$ be a correct term such that $t' \ltimes \mathsf{src}(\rho)$. Then $\rho/t' \equiv \sigma/t'$.*

Proof. This is an immediate consequence of Lemma 4.16, since ρ/t' and σ/t' are both complete developments of $\mathcal{M}/t'$, hence permutation equivalent. □

Proposition A.25 (Compatibility). *Let $\rho \equiv \sigma$ be permutation equivalent derivations in the λ-calculus, and let $t' \in \mathcal{T}^{\#}$ be a correct term such that $t' \ltimes \mathsf{src}(\rho)$. Then:*

1. $t'/\rho = t'/\sigma$
2. $\rho/t' \equiv \sigma/t'$

Proof. Recall that permutation equivalence is defined as the reflexive and transitive closure of the permutation axiom $\tau_1 R\sigma\tau_2 \equiv \tau_1 S\rho\tau_2$, where τ_1 and τ_2 are arbitrary derivations, σ is a complete development of S/R, and ρ is a complete development of R/S. We prove each item separately:

1. For item 1., we proceed by induction on the derivation that $\rho \equiv \sigma$. Reflexivity and transitivity are trivial, so we concentrate on the permutation axiom itself. The interesting case is the permutation axiom. Let $\tau_1 R\sigma\tau_2 \equiv \tau_1 S\rho\tau_2$, where τ_1 and τ_2 are arbitrary derivations, σ is a complete development of S/R, and ρ is a complete development of R/S, and let us show that $t'/\tau_1 R\sigma\tau_2 = t'/\tau_1 S\rho\tau_2$. By Lemma 4.14 we have that:

$$t'/\tau_1 R\sigma\tau_2 = ((t'/\tau_1)/R\sigma)/\tau_2 \qquad \text{and} \qquad t'/\tau_1 S\rho\tau_2 = ((t'/\tau_1)/S\rho)/\tau_2$$

so, without loss of generality, it suffices to show that the following holds for an arbitrary term $s' \in \mathcal{T}^{\#}$:

$$s'/R\sigma = s'/S\rho$$

Note that, by definition of simulation residual, the derivations below have the indicated sources and targets:

$$\begin{array}{rcl} R\sigma/s' & : & s' \to s'/R\sigma \\ S\rho/s' & : & s' \to s'/S\rho \end{array}$$

Moreover:

$$\begin{array}{rcll} R\sigma/s' & = & (R/s')(\sigma/(s'/R)) & \\ & \equiv & (R/s')((S/R)/(s'/R)) & \text{by Lemma A.24} \\ & \equiv & (R/s')((S/s')/(R/s')) & \text{by the basic cube lemma (Lemma 4.15)} \\ & \equiv & (S/s')((R/s')/(S/s')) & \text{since } A(B/A) \equiv B(A/B) \text{ holds in general} \\ & \equiv & (S/s')((R/S)/(s'/S)) & \text{by the basic cube lemma (Lemma 4.15)} \\ & = & (S/s')(\rho/(s'/S)) & \text{by Lemma A.24} \\ & = & S\rho/s' & \end{array}$$

So $R\sigma/s'$ and $S\rho/s'$ are permutation equivalent. In particular, they have the same target, so $s'/R\sigma = s'/S\rho$ as required.

2. The proof of item 2. is also by induction on the derivation that $\rho \equiv \sigma$. Let $\tau_1 R\sigma\tau_2 \equiv \tau_1 S\rho\tau_2$, where τ_1 and τ_2 are arbitrary derivations, σ is a complete development of S/R, and ρ is a complete development of R/S, and let us show that $\tau_1 R\sigma\tau_2/t' = \tau_1 S\rho\tau_2/t'$. By Lemma 4.14 we have that:

$$\tau_1 R\sigma\tau_2/t' = (\tau_1/t')(R\sigma/s')(\tau_2/u') \quad \text{and} \quad \tau_1 S\rho\tau_2/t' = (\tau_1/t')(S\rho/s')(\tau_2/u'')$$

where $s' = t'/\tau_1$, $u' = s'/\tau_1 R\sigma$, and $u'' = s'/\tau_1 S\rho$.
In a manner similar as before, we can prove that $R\sigma/s' \equiv S\rho/s'$. And, with that and Item 1, we can see that $u' = u''$, hence $(\tau_2/u') \equiv (\tau_2/u'')$, which finishes the proof.

□

A.18 Proofs from Section 5.2 — Sieving

Proof of Lemma 5.9 — Sieving is well-defined

The operation $\rho \downarrow t'$ is well-defined.
Proof. Proving this amounts to showing that the recursion scheme is well-founded. It suffices to show that there is a measure M ranging over the non-negative integers such that $M(\rho, t') > M(\rho/R_0, t'/R_0)$ whenever R_0 is the leftmost coarse step for (ρ, t'). Indeed, if we define:

$$M(\rho, t') \stackrel{\text{def}}{=} |\rho/t'|$$

where $|\rho/t'|$ stands for the length of the derivation ρ/t' in the distributive lambda-calculus, then we may conclude by checking that the following inequality holds:

$$|\rho/t'| > |(\rho/R_0)/(t'/R_0)|$$

First observe that $R_0 \sqsubseteq \rho$ so, by Corollary 4.20, $R_0/t' \sqsubseteq \rho/t'$ and, by Proposition 3.16, $\mathsf{names}(R_0/t') \sqsubseteq \mathsf{names}(\rho/t')$. Then we have:

$\lvert\rho/t'\rvert$	$=$	$\#\mathsf{names}(\rho/t')$	by Corollary 3.14
	$>$	$\#(\mathsf{names}(\rho/t') \setminus \mathsf{names}(R_0/t'))$	since $R_0/t' \neq \varnothing$ and $\mathsf{names}(R_0/t') \subseteq \mathsf{names}(\rho/t')$
	$=$	$\#\mathsf{names}((\rho/t')/(R_0/t'))$	by Lemma 3.15
	$=$	$\#\mathsf{names}((\rho/R_0)/(t'/R_0))$	by Corollary 3.17 and Lemma 4.19
	$=$	$\lvert(\rho/R_0)/(t'/R_0)\rvert$	by Corollary 3.14

□

Proof of Lemma A.34 — Sieving is compatible with permutation equivalence

Let $\rho \equiv \sigma$. Then $\rho \downarrow t' \equiv \sigma \downarrow t'$.
Proof. First observe that, given two permutation equivalent derivations ρ and σ, a step R is coarse for (ρ, t') if and only if R is coarse for (σ, t'), since:

$$(R \sqsubseteq \rho) \iff (R/\rho = \varnothing) \iff (R/\sigma = \varnothing) \iff (R \sqsubseteq \sigma)$$

We proceed by induction on the length of $\rho \downarrow t'$. There are two cases, depending on whether there is a coarse step for (ρ, t').

1. **If there are no coarse steps for** (ρ, t')**.** Then there are no coarse steps for (σ, t'), so $\rho \downarrow t' = \epsilon = \sigma \downarrow t'$.
2. **If there exists a coarse step for** (ρ, t')**.** Let R_0 be the leftmost coarse step for (ρ, t'). By the preceding observation, R_0 is also the leftmost coarse step for (σ, t').
Note also that $\rho/R_0 \equiv \sigma/R_0$, as a consequence of the well-known properties of permutation equivalence. Moreover, the length of $(\rho/R_0) \downarrow (t'/R_0)$ is shorter than the length of $\rho \downarrow t'$, so by *i.h.*, $(\rho/R_0) \downarrow (t'/R_0) = (\sigma/R_0) \downarrow (t'/R_0)$. To conclude the proof, note that:

$$\rho \downarrow t' = R_0((\rho/R_0) \downarrow (t'/R_0)) \overset{\text{h.i.}}{=} R_0((\sigma/R_0) \downarrow (t'/R_0)) = \sigma \downarrow t'$$

□

A.19 Proofs from Section 5.3 — Some properties

Lemma A.26 (Different redexes have disjoint simulation residuals)**.** *If* $R \neq S$*, then* $\mathsf{names}(R/t')$ *and* $\mathsf{names}(S/t')$ *are disjoint.*

Proof. Let $R : \mathtt{C}\langle(\lambda x.r)s\rangle \to \mathtt{C}\langle r\{x := s\}\rangle$. We proceed by induction on $\mathtt{C}$.

1. $\mathtt{C} = \Box$**.** We have that $R : (\lambda x.r)s \to r\{x := s\}$. Furthermore, $t' = (\lambda^{\ell} x.r')\vec{s}$ and $t'/R = r'\{x := \vec{s}\}$, hence $\mathsf{names}(R/t') = \{\ell\}$. Now there are two cases depending on where S is located.
   1.1 S **is in** r**.** So $r = \mathtt{C}_r\langle(\lambda y.u)v\rangle$. Now, $S : (\lambda x.\mathtt{C}_r\langle(\lambda y.u)v\rangle)s \to (\lambda x.\mathtt{C}_r\langle u\{y := v\}\rangle)s$. That implies that $t' = (\lambda^{\ell} x.\mathtt{C}'_r\langle(\lambda^{\ell_1} y.u_1)v_1, \ldots, (\lambda^{\ell_n} y.u_n)v_n\rangle)\vec{s}$, and, in turn, the simulated step yields $t'/S = (\lambda^{\ell} x.\mathtt{C}'_r\langle u_1\{y := v_1\}, \ldots, u_n\{y := v_n\}\rangle)\vec{s}$, so $\mathsf{names}(S/t') = \{\ell_1, \ldots, \ell_n\}$, which does not contain ℓ because labels are pairwise distinct.
   1.2 S **is in** s**.** So $s = \mathtt{C}_s\langle(\lambda y.u)v\rangle$. Like before, $S : (\lambda x.r)\mathtt{C}_s\langle(\lambda y.u)v\rangle \to (\lambda x.r)\mathtt{C}_s\langle u\{y := v\}\rangle$. By Lemma A.14 we have that $t' = (\lambda^{\ell} x.r')[\mathtt{C}'_s\langle(\lambda^{\ell_{i,1}} y.u_{i,1})v_{i,1}, \ldots, (\lambda^{\ell_{i,m_i}} y.u_{i,m_i})v_{i,m_i}\rangle]_{i=1}^{n}$, then $\mathsf{names}(S/t') = \bigcup_{i=1}^{n}\{\ell_{i,1}, \ldots, \ell_{i,m_i}\}$, which does not contain ℓ because labels are pairwise distinct.
2. $\mathtt{C} = \lambda z.\mathtt{C}_1$**.** This case is straightforward by inductive hypothesis.
3. $\mathtt{C} = p\ \mathtt{C}_1$**.** Now we have that $R : p\ \mathtt{C}_1\langle(\lambda y.u)v\rangle \to p\ \mathtt{C}_1\langle u\{y := v\}\rangle$. And we can again separate in cases depending on where S is.
   3.1 S **is in** p**.** What that means is that $S : p\ \mathtt{C}_1\langle(\lambda y.u)v\rangle \to q\ \mathtt{C}_1\langle(\lambda y.u)v\rangle$. Using the same reasoning as before, we know that $t' = p'[\mathtt{C}'_{1,i}\langle(\lambda^{\ell_{i,1}} y.u_{i,1})v_{i,1}, \ldots, (\lambda^{\ell_{i,m_i}} y.u_{i,m_i})v_{i,m_i}\rangle]_{i=1}^{n}$. As S/t' only reduces lambdas in p', and S/t' only reduces lambdas on the argument list of the application, the set of names are disjoint.
   3.2 S **is in** $\mathtt{C}_1\langle(\lambda y.u)v\rangle$**.** This case is straightforward by inductive hypothesis.
4. $\mathtt{C} = \mathtt{C}_1\ p$**.** In this case $R : \mathtt{C}_1\langle\lambda y.u)v\rangle\ p \to \mathtt{C}_1\langle u\{y := v\}\rangle\ p$, and we will, once again, divide in two cases, depending on where the redex S is located.
   4.1 S **is in** $\mathtt{C}_1\langle(\lambda y.u)v\rangle$**.** This case is straightforward by inductive hypothesis.
   4.2 S **is in** p**.** More precisely, what we have is that $S : \mathtt{C}_1\langle(\lambda y.u)v\rangle\ p \to \mathtt{C}_1\langle(\lambda y.u)v\rangle\ q$. Moreover, $t' = \mathtt{C}_1\langle(\lambda^{\ell_1} y.u_1)v_1, \ldots, (\lambda^{\ell_m} y.u_m)v_m\rangle[p_1, \ldots, p_n]$. Hence, $\mathsf{names}(R/t') = \{\ell_1, \ldots, \ell_m\}$. Also, S/t' only reduces lambdas in the terms p_1, ..., p_n, which must be different than the ones reduced by R/t' because lambda labels are pairwise distinct.

□

A set of steps and a disjoint derivation have disjoint simulation residuals

Lemma A.27 (A set of steps and a disjoint derivation have disjoint simulation residuals)**.** *Let $\mathcal{M}$ be a set of coinitial steps such that no residuals of any step in $\mathcal{M}$ are contracted along a derivation σ. Then* $\mathsf{names}(\mathcal{M}/t')$ *and* $\mathsf{names}(\sigma/t')$ *are disjoint.*

Proof. We proceed by induction on σ. If σ is empty, it is immediate, so suppose that $\sigma = T\tau$. Note that $T \notin \mathcal{M}$, since residuals of redexes in $\mathcal{M}$ are never contracted in the derivation $T\tau$. This implies that $\mathsf{names}(\mathcal{M}/t') \cap \mathsf{names}(T/t') = \varnothing$ by Lemma A.27. Moreover, this means that:

$$\begin{array}{rcll} \mathsf{names}(\mathcal{M}/t') & = & \mathsf{names}(\mathcal{M}/t') \setminus \mathsf{names}(T/t') & \text{since they are disjoint sets} \\ & = & \mathsf{names}((\mathcal{M}/t')/(T/t')) & \text{by Lemma 3.15} \\ & = & \mathsf{names}((\mathcal{M}/T)/(t'/T)) & \text{by Corollary 3.17 and Lemma 4.19} \end{array}$$

Then:

$$\begin{array}{rcl} \mathsf{names}(\mathcal{M}/t') \cap \mathsf{names}(\sigma/t') & = & \mathsf{names}(\mathcal{M}/t') \cap \mathsf{names}(T\tau/t') \\ & = & \mathsf{names}(\mathcal{M}/t') \cap \mathsf{names}((T/t')(\tau/(t'/T))) \\ & = & (\mathsf{names}(\mathcal{M}/t') \cap \mathsf{names}(T/t')) \cup \\ & & (\mathsf{names}(\mathcal{M}/t') \cap \mathsf{names}(\tau/(t'/T))) \\ & = & \mathsf{names}(\mathcal{M}/t') \cap \mathsf{names}(\tau/(t'/T)) \\ & & \text{since } \mathsf{names}(\mathcal{M}/t') \cap \mathsf{names}(T/t') \text{ is empty} \\ & = & \mathsf{names}((\mathcal{M}/T)/(t'/T)) \cap \mathsf{names}(\tau/(t'/T)) \\ & = & \varnothing \quad \text{by } \textit{i.h.} \end{array}$$

To justify the last step, observe that residuals of redexes in the set $\mathcal{M}/T$ may not be contracted along the derivation τ, since this would imply that a residual of a redex in the set $\mathcal{M}$ is contracted along the derivation $T\tau$, contradicting the hypothesis. □

Proof of Proposition 5.14 — Characterization of garbage

Let $\rho : t \twoheadrightarrow_\beta s$ and $t' \ltimes t$. The following are equivalent:

1. $\rho \downarrow t' = \epsilon$.
2. There are no coarse steps for (ρ, t').
3. The derivation ρ is t'-garbage.

Proof. It is immediate to check that items 1 and 2 are equivalent, by definition of sieving, so let us prove 2 $\implies$ 3 and 3 $\implies$ 2:

- (2 $\implies$ 3) We prove the contrapositive, namely that if ρ is not garbage, there is a coarse step for (ρ, t'). Suppose that ρ is not garbage, *i.e.* $\rho/t' \neq \epsilon$. Then by Proposition 5.4 we have that ρ can be written as $\rho = \rho_1 R \rho_2$ where all the steps in ρ_1 are garbage and R is not garbage. By the fact that garbage only creates garbage (Lemma 5.12) the step R has an ancestor R_0, *i.e.* $R \in R_0/\rho_1$. Moreover, since garbage only duplicates garbage (Lemma 5.13) we have that $R_0/\rho_1 = R$. Given that R is not garbage, we have that:

$$\begin{array}{rcll} (R_0/t')/(\rho_1/t') & = & (R_0/\rho_1)/(t'/\rho_1) & \text{by Lemma 4.19} \\ & = & R/(t'/\rho_1) & \\ & \neq & \varnothing & \end{array}$$

  Since $(R_0/t')/(\rho_1/t') \neq \varnothing$, in particular, $R_0/t' \neq \varnothing$, which means that R_0 is not garbage. Moreover, $R_0 \sqsubseteq \rho_1 R \rho_2 = \rho$. So R_0 is coarse for (ρ, t').

- $(3 \implies 2)$ Let ρ be garbage, suppose that there is a coarse step R for (ρ, t'), and let us derive a contradiction. Since R is coarse for (ρ, t'), we have that $R \sqsubseteq \rho$, so $R/t' \sqsubseteq \rho/t'$ by Corollary 4.20. But ρ/t' is empty because ρ is t'-garbage, that is, $R/t' \sqsubseteq \sigma/t' = \epsilon$, which means that R is also t'-garbage. This contradicts the fact that R is coarse for (ρ, t').

□

Proof of Proposition 5.15 — Characterization of garbage-free derivations

Let $\rho : t \twoheadrightarrow_\beta s$ and $t' \ltimes t$. The following are equivalent:

1. ρ is t'-garbage-free.
2. $\rho \equiv \rho \downarrow t'$.
3. $\rho \equiv \sigma \downarrow t'$ for some derivation σ.

Proof. Let us prove $1 \implies 2 \implies 3 \implies 1$:

- $(1 \implies 2)$ Suppose that ρ is t'-garbage-free, and let us show that $\rho \equiv \rho \downarrow t'$ by induction on the length of $\rho \downarrow t'$.

  If there are no coarse steps for (ρ, t'), By Proposition 5.14, any derivation with no coarse steps is garbage. So ρ is t'-garbage. Since ρ is garbage-free, this means that $\rho = \epsilon$. Hence $\rho = \epsilon = \rho \downarrow t'$, as required.

  If there exists a coarse step for (ρ, t'), let R_0 be the leftmost such step. Note that $\rho \equiv R_0(\rho/R_0)$ since $R_0 \sqsubseteq \rho$. Moreover, we claim that ρ/R_0 is (t'/R_0)-garbage-free. Let $\sigma \sqsubseteq \rho/R_0$ such that $(\rho/R_0)/\sigma$ is garbage with respect to the term $(t'/R_0)/\sigma = t'/R_0\sigma$, and let us show that $(\rho/R_0)/\sigma$ is empty. Note that:

$$\begin{array}{lll} R_0\sigma \sqsubseteq R_0(\rho/R_0) & \text{since } \sigma \sqsubseteq \rho/R_0 \\ \equiv \rho & \text{as already noted} \end{array}$$

  Moreover, we know that the derivation $\rho/R_0\sigma = (\rho/R_0)/\sigma$ is $(t'/R_0\sigma)$-garbage. So, given that ρ is t'-garbage-free, we conclude that $\rho/R_0\sigma = \epsilon$, that is $(\rho/R_0)/\sigma = \epsilon$, which completes the proof of the claim that ρ/R_0 is (t'/R_0)-garbage-free. We conclude as follows:

$$\begin{array}{lll} \rho \equiv R_0(\rho/R_0) & \text{as already noted} \\ \equiv R_0((\rho/R_0) \downarrow (t'/R_0)) & \text{by } \textit{i.h.} \text{ since } \rho/R_0 \text{ is } (t'/R_0)\text{-garbage-free} \\ \equiv \rho \downarrow t' & \text{by definition of sieving} \end{array}$$

- $(2 \implies 3)$ Obvious, taking $\sigma := \rho$.

- $(3 \implies 1)$ Let $\rho \equiv \sigma \downarrow t'$. Let us show that ρ is garbage-free by induction on the length of $\sigma \downarrow t'$.

  If there are no coarse steps for (σ, t'), then $\rho \equiv \sigma \downarrow t' = \epsilon$, which means that $\rho = \epsilon$. Observe that the empty derivation is trivially garbage-free.

  If there exists a coarse step for (σ, t'), let R_0 be the leftmost such step. Then $\rho \equiv \sigma \downarrow t' = R_0((\sigma/R_0) \downarrow (t'/R_0))$. To see that ρ is t'-garbage-free, let $\tau \sqsubseteq \rho$ such that ρ/τ is garbage, and let us show that ρ/τ is empty. We know that ρ/τ is of the following form (modulo permutation equivalence):

$$\rho/\tau \equiv \frac{R_0((\sigma/R_0) \downarrow (t'/R_0))}{\tau} = \left(\frac{R_0}{\tau}\right)\left(\frac{(\sigma/R_0) \downarrow (t'/R_0)}{\tau/R_0}\right)$$

That is, we know that the following derivation is (t'/τ)-garbage, and it suffices to show that it is empty:

$$\left(\frac{R_0}{\tau}\right)\left(\frac{(\sigma/R_0)\downarrow(t'/R_0)}{\tau/R_0}\right)$$

Recall that, in general, AB is garbage if and only if A and B are garbage (Proposition 5.4). Similarly, AB is empty if and only if A and B are empty. So it suffices to prove the two following implications:

(**A**)	If	R_0/τ is garbage,	then it is empty.
(**B**)	If	$((\sigma/R_0)\downarrow(t'/R_0))/(\tau/R_0)$ is garbage,	then it is empty.

Let us check that each implication holds:

- **(A)** Suppose that R_0/τ is (t'/τ)-garbage, and let us show that R_0/τ is empty. Knowing that the derivation R_0/τ is garbage means that $(R_0/\tau)/(t'/\tau) = \varnothing$. Since R_0 is the leftmost step coarse for (σ, t'), by Lemma A.30 we have that $\#(R_0/\tau) \leq 1$. If R_0/τ is empty we are done, since this is what we wanted to prove.

  The remaining possibility is that R_0/τ be a singleton. We argue that this case is impossible. Note that for every prefix $\tau_1 \sqsubseteq \tau$, the set R_0/τ_1 is also a singleton, since otherwise it would be empty, as a consequence of Lemma A.30. So we may apply Lemma A.31 and conclude that, since R_0 is not t'-garbage then R_0/τ is not (t'/τ)-garbage. This contradicts the hypothesis.
- **(B)** Suppose that $((\sigma/R_0)\downarrow(t'/R_0))/(\tau/R_0)$ is garbage with respect to the term $(t'/R_0)/(\tau/R_0)$, and let us show that it is empty. Since $(\sigma/R_0)\downarrow(t'/R_0)$ is a shorter derivation than $\sigma\downarrow t'$, we may apply the *i.h.* we obtain that $(\sigma/R_0)\downarrow(t'/R_0)$ is (t'/R_0)-garbage-free. Moreover, the following holds:

  $$\tau/R_0 \sqsubseteq \rho/R_0 \equiv (\sigma/R_0)\downarrow(t'/R_0)$$

  So, by definition of $(\sigma/R_0)\downarrow(t'/R_0)$ being garbage-free, the fact that the derivation $((\sigma/R_0)\downarrow(t'/R_0))/(\tau/R_0)$ is garbage implies that it is empty, as required.

□

Proof of Proposition 5.16 — Properties of sieving

To prove Proposition 5.16 we first prove various auxiliary results.

Lemma A.28 (The sieve is a prefix)**.** *Let $\rho : t \twoheadrightarrow_\beta s$ and $t' \ltimes t$. Then $\rho\downarrow t' \sqsubseteq \rho$.*

Proof. By induction on the length of $\rho\downarrow t'$. There are two cases, depending on whether there exists a coarse step for (ρ, t').

1. **If there are no coarse steps for** (ρ, t')**.** Then trivially $\rho\downarrow t' = \epsilon \sqsubseteq \rho$.
2. **If there exists a coarse step for** (ρ, t')**.** Let ρ_0 be the leftmost coarse step for (ρ, t'). Then:

$$\begin{array}{lcll} \rho\downarrow t' & = & R_0((\rho/R_0)\downarrow(t'/R_0)) & \\ & \sqsubseteq & R_0(\rho/R_0) & \text{by } \textit{i.h.} \\ & \equiv & \rho(R_0/\rho) & \text{since } A(B/A) \equiv B(A/B) \text{ in general} \\ & = & \rho & \text{since } R_0 \sqsubseteq \rho \text{ as } R_0 \text{ is coarse for } (\rho, t') \end{array}$$

□

Lemma A.29 (Garbage only interacts with garbage)**.** *The following hold:*

1. **Garbage only creates garbage.** *Let R and S be composable steps in the λ-calculus, and let $t' \ltimes \mathsf{src}(R)$. If R creates S and R is t'-garbage, then S is (t'/R)-garbage.*
2. **Garbage only duplicates garbage.** *Let R and S be coinitial steps in the λ-calculus and let $t' \ltimes \mathsf{src}(R)$. If R duplicates S,* i.e. *$\#(S/R) > 1$, and R is t'-garbage, then S is (t'/R)-garbage.*

Proof. We prove each item separately:

1. According to Lévy [Lév78], there are three creation cases in the λ-calculus. We consider the three possibilities for R creating S:
   - 1.1 **Case I,** $\mathtt{C}\langle(\lambda x.x)(\lambda y.s)u\rangle \xrightarrow{R}_\beta \mathtt{C}\langle(\lambda y.s)u\rangle \xrightarrow{S}_\beta \mathtt{C}\langle s\{y := u\}\rangle$**.** Then by Lemma A.14, the term t' is of the form $\mathtt{C}'\langle\Delta_1,\dots,\Delta_n\rangle$ where $\mathtt{C}'$ is an n-hole context such that $\mathtt{C}' \ltimes \mathtt{C}\langle\Box\, u\rangle$ and $\Delta_i \ltimes (\lambda x.x)(\lambda y.s)$ for all $1 \le i \le n$. Since R is garbage, we know that actually $n = 0$. So $t' \ltimes \mathtt{C}\langle\Box\, u\rangle$ and $R/t' : t' \twoheadrightarrow_\# t' = t'/R$ in zero steps. Hence $t' \ltimes \mathtt{C}\langle(\lambda y.s)u\rangle$, so by Lemma A.14, the term t' can be written in a unique way as $\mathtt{C}''\langle\Sigma_1,\dots,\Sigma_m\rangle$, where $\mathtt{C}''$ is an m-hole context such that $\mathtt{C}'' \ltimes \mathtt{C}\langle\Box\, u\rangle$ and $\Sigma_i \ltimes \lambda y.s$ for all $1 \le i \le m$. Since the decomposition is unique and $t' \ltimes \mathtt{C}\langle\Box\, u\rangle$, we conclude that $m = 0$. Hence S is (t'/R)-garbage.
   - 1.2 **Case II,** $\mathtt{C}\langle(\lambda x.\lambda y.s)\, u\, r\rangle \xrightarrow{R}_\beta \mathtt{C}\langle(\lambda y.s\{x := u\})\, r\rangle \xrightarrow{S}_\beta \mathtt{C}\langle s\{x := u\}\{y := r\}\rangle$**.** Then by Lemma A.14, the term t' is of the form $\mathtt{C}'\langle\Delta_1,\dots,\Delta_n\rangle$ where $\mathtt{C}'$ is an n-hole context such that $\mathtt{C}' \ltimes \mathtt{C}\langle\Box\, r\rangle$ and $\Delta_i \ltimes (\lambda x.\lambda y.s)\, u$ for all $1 \le i \le n$. Since R is garbage, we know that actually $n = 0$. So $t' \ltimes \mathtt{C}\langle\Box\, r\rangle$ and $R/t' : t' \twoheadrightarrow_\# t' = t'/R$ in zero steps. Hence $t' \ltimes \mathtt{C}\langle(\lambda y.s\{x := u\})\, r\rangle$, so by Lemma A.14, the term t' can be written in a unique way as $\mathtt{C}''\langle\Sigma_1,\dots,\Sigma_n\rangle$, where $\mathtt{C}''$ is an m-hole context such that $\mathtt{C}'' \ltimes \mathtt{C}\langle\Box\, r\rangle$ and $\Sigma_i \ltimes \lambda y.s\{x := u\}$ for all $1 \le i \le m$. Since the decomposition is unique and $t' \ltimes \mathtt{C}\langle\Box\, r\rangle$, we conclude that $m = 0$. Hence S is (t'/R)-garbage.
   - 1.3 **Case III,** $\mathtt{C}_1\langle(\lambda x.\mathtt{C}_2\langle x\, s\rangle)\,(\lambda y.u)\rangle \xrightarrow{R}_\beta \mathtt{C}_1\langle\hat{\mathtt{C}}_2\langle(\lambda y.u)\,\hat{s}\rangle\rangle \xrightarrow{S}_\beta \mathtt{C}_1\langle\hat{\mathtt{C}}_2\langle u\{y := \hat{s}\}\rangle\rangle$**, where** $\hat{\mathtt{C}}_2 = \mathtt{C}_2\{x := \lambda y.u\}$ **and** $\hat{t} = t\{x := \lambda y.u\}$**.** Then by Lemma A.14, the term t' is of the form $\mathtt{C}'\langle\Delta_1,\dots,\Delta_n\rangle$ where $\mathtt{C}'$ is an n-hole context such that $\mathtt{C}' \ltimes \mathtt{C}_1$ and $\Delta_i \ltimes (\lambda x.\mathtt{C}_2\langle x\, s\rangle)\,(\lambda y.u)$ for all $1 \le i \le n$. Since R is garbage, we know that actually $n = 0$. So $t' \ltimes \mathtt{C}_1$ and $R/t' : t' \twoheadrightarrow_\# t' = t'/R$ in zero steps. Hence $t' \ltimes \mathtt{C}_1\langle\hat{\mathtt{C}}_2\langle(\lambda y.u)\,\hat{s}\rangle\rangle$, so by Lemma A.14, the term t' can be written in a unique way as $\mathtt{C}''\langle\Sigma_1,\dots,\Sigma_m\rangle$, where $\mathtt{C}'' \ltimes \mathtt{C}_1$ and $\Sigma_i \ltimes \hat{\mathtt{C}}_2\langle(\lambda y.u)\,\hat{s}\rangle$ for all $1 \le i \le m$. Since the decomposition is unique and $t' \ltimes \mathtt{C}_1$, we conclude that $m = 0$. Hence S is (t'/R)-garbage.
2. Since R duplicates S, the redex contracted by S lies inside the argument of R, that is, the source term is of the form $\mathtt{C}_1\langle(\lambda x.t)\mathtt{C}_2\langle(\lambda y.s)u\rangle\rangle$ where the pattern of R is $(\lambda x.t)\mathtt{C}_2\langle(\lambda y.s)u\rangle$, and the pattern of S is $(\lambda y.s)u$. By Lemma A.14, the term t' is of the form $\mathtt{C}'\langle\Delta_1,\dots,\Delta_n\rangle$ where $\mathtt{C}'$ is an n-hole context such that $\mathtt{C}' \ltimes \mathtt{C}$ and $\Delta_i \ltimes (\lambda x.t)\mathtt{C}_2\langle(\lambda y.s)u\rangle$ for all $1 \le i \le n$. Since R is garbage, we know that $n = 0$. By Lemma A.14, the term t' can be written as $\mathtt{C}''\langle\Sigma_1,\dots,\Sigma_m\rangle$ where $\mathtt{C}'' \ltimes \mathtt{C}_1\langle(\lambda x.t)\mathtt{C}_2\rangle$ and $\Sigma_i \ltimes (\lambda y.s)u$ for all $1 \le i \le m$. Note that $t' \ltimes \mathtt{C}_1$ so $t' \ltimes \mathtt{C}_1\langle(\lambda x.t)\mathtt{C}_2\rangle$, as can be checked by induction on $\mathtt{C}_1$. Since the decomposition is unique, this means that $m = 0$, and thus S is garbage.

□

Lemma A.30 (The leftmost coarse step has at most one residual)**.** *Let R_0 be the leftmost coarse step for (ρ, t'), and let $\sigma \sqsubseteq \rho$. Then $\#(R_0/\sigma) \le 1$.*

Proof. By induction on the length of σ. The base case is immediate. For the inductive step, let $\sigma = S\tau \sqsubseteq \rho$. Then in particular $S \sqsubseteq \rho$. We consider two cases, depending on whether $R_0 = S$.

1. **Equal,** $R_0 = S$**.** Then $R_0/\sigma = R_0/R_0\tau = \varnothing$ and we are done.
2. **Non-equal,** $R_0 \neq S$**.** First we argue that R_0/S has exactly one residual $R_1 \in R_0/S$. To see this, we consider two further cases, depending on whether S is t'-garbage or not:
   2.1 **If** S **is not** t'**-garbage.** Then $S \sqsubseteq \rho$ and $S/t' = \varnothing$, so S is coarse for (ρ, t'). Since R_0 is the leftmost coarse step, this means that R is to the left of S. So R_0 has exactly one residual $R_1 \in R_0/S$.
   2.2 **If** S **is** t'**-garbage.** Let us write the term t as $t = \mathtt{C}\langle(\lambda x.s)u\rangle$, where $(\lambda x.s)u$ is the pattern of the redex S. By Lemma A.14 the term t' is of the form $t' = \mathtt{C}'[\Delta_1, \ldots, \Delta_n]$, where $\mathtt{C}'$ is a many-hole context such that $\mathtt{C}' \ltimes \mathtt{C}$ and $\Delta_i \ltimes (\lambda x.s)u$ for all $1 \leq i \leq n$. We know that S is garbage, so $n = 0$ and $t' = \mathtt{C}'$ is actually a 0-hole context (*i.e.* a term). On the other hand, R_0 is coarse for (ρ, t'), so in particular it is not t'-garbage. This means that the pattern of the redex R_0 cannot occur inside the argument u of the redex S. So S does not erase or duplicate R, *i.e.* R_0 has exactly one residual $R_1 \in R_0/S$.

   Now we have that $R_0/S\tau = R_1/\tau$. We are left to show that $\#(R_0/S\tau) \leq 1$. Let us show that we may apply the *i.h.* on R_1. More precisely, observe that $\tau \sqsubseteq \rho/S$ since $S\tau \sqsubseteq \rho$. To apply the *i.h.* it suffices to show that R_1 is coarse for $(\rho/S, t'/S)$. Indeed, we may check the two conditions for the definition that R_1 is coarse for $(\rho/S, t'/S)$.
   2.1 Firstly, $R_1 = R_0/S \sqsubseteq \rho/S$ holds, as a consequence of the fact that $R_0 \sqsubseteq \rho$.
   2.2 Secondly, we may check that R_1 is not (t'/S)-garbage. To see this, *i.e.* that $R_1/(t'/S)$ is non-empty, we check that $\mathsf{names}(R_1/(t'/S))$ is non-empty.

$$\begin{array}{lcll} \mathsf{names}(R_1/(t'/S)) & = & \mathsf{names}((R_0/S)/(t'/S)) & \text{by definition of } R_1 \\ & = & \mathsf{names}((R_0/t')/(S/t')) & \text{by Lemma 4.19} \\ & = & \mathsf{names}(R_0/t') \setminus \mathsf{names}(S/t') & \text{by Lemma 3.15} \\ & = & \mathsf{names}(R_0/t') & \end{array}$$

   For the last step, note that $\mathsf{names}(R_0/t')$ and $\mathsf{names}(S/t')$ are disjoint, since $R_0 \neq S$. Applying the *i.h.*, we obtain that $\#(R_0/S\tau) = \#(R_1/\tau) \leq 1$.

□

Lemma A.31. *Let R be a step, let ρ a coinitial derivation, and let $t' \ltimes \mathsf{src}(R)$. Suppose that R is not t'-garbage, and that R/ρ_1 is a singleton for every prefix $\rho_1 \sqsubseteq \rho$. Then R/ρ is not (t'/ρ)-garbage.*

Proof. By induction on ρ. The base case, when $\rho = \epsilon$, is immediate since we know that R is not garbage. For the inductive step, suppose that $\rho = S\sigma$. We know that R/S is a singleton, so let $R_1 = R/S$. Note that $R_1/(t'/S) = (R/S)/(t'/S) = (R/t')/(S/t')$ by Lemma 4.19. We know that R/t' is non-empty, because R is not garbage. Moreover, $\mathsf{names}(R/t')$ and $\mathsf{names}(S/t')$ are disjoint since $R \neq S$. So $\#\mathsf{names}((R/t')/(S/t')) = \mathsf{names}(R/t')$ by Lemma 3.15. This means that the set $R_1/(t'/S)$ is non-empty, so R_1 is not (t'/S)-garbage . By *i.h.* we obtain that R_1/σ is not $((t'/S)/\sigma)$-garbage, which means that $(R/S\sigma)/(t'/S\sigma) \neq \varnothing$, *i.e.* that $R/S\sigma$ is not garbage, as required. To be able to apply the *i.h.*, observe that if σ_1 is a prefix of σ, then $S\sigma_1$ is a prefix of ρ, so the fact that R has a single residual after $S\sigma_1$ implies that the step $R_1 = R/S$ has a single residual after σ_1. □

Lemma A.32 (The projection after a sieve is garbage)**.** *Let $\rho : t \twoheadrightarrow_\beta s$ and $t' \ltimes t$. Then $\rho/(\rho \downarrow t')$ is $(t'/(\rho \downarrow t'))$-garbage.*

Proof. By induction on the length of $\rho \downarrow t'$.

If there are no coarse steps for (ρ, t'), then $\rho \downarrow t' = \epsilon$. Moreover, by Proposition 5.14, a derivation with no coarse steps is garbage. So $\rho/(\rho \downarrow t') = \rho$ is garbage, as required.

If there exists a coarse step for (ρ, t'), let R_0 be the leftmost such step, and let $\sigma = \rho/R_0$ and $s' = t'/R_0$. Then:

$$\begin{array}{rcll} \rho/(\rho \downarrow t') & = & \rho/(R_0((\rho/R_0) \downarrow (t'/R_0)) & \text{by definition} \\ & = & (\rho/R_0)/((\rho/R_0) \downarrow (t'/R_0)) & \\ & = & \sigma/(\sigma \downarrow s') & \end{array}$$

By *i.h.*, $\sigma/(\sigma \downarrow s')$ is $(s'/(\sigma \downarrow s'))$-garbage. That is:

$$\frac{\sigma/(\sigma \downarrow s')}{s'/(\sigma \downarrow s')} = \epsilon$$

Unfolding the definitions of σ and s' we have that:

$$\frac{(\rho/R_0)/((\rho/R_0) \downarrow (t'/R_0))}{(t'/R_0)/((\rho/R_0) \downarrow (t'/R_0))} = \epsilon$$

Equivalently:

$$\frac{\rho/R_0((\rho/R_0) \downarrow (t'/R_0))}{t'/R_0((\rho/R_0) \downarrow (t'/R_0))} = \epsilon$$

Finally, by definition of sieve,

$$\frac{\rho/(\rho \downarrow t')}{t'/(\rho \downarrow t')} = \epsilon$$

which means that $\rho/(\rho \downarrow t')$ is $(t'/(\rho \downarrow t'))$-garbage, as required. □

Lemma A.33 (Sieving trailing garbage)**.** *Let ρ and σ be composable derivations, and let $t' \ltimes \mathsf{src}(\rho)$. If σ is (t'/ρ)-garbage, then $\rho\sigma \downarrow t' = \rho \downarrow t'$.*

Proof. By induction on the length of $\rho \downarrow t'$. There are two cases, depending on whether there exists a coarse step for (ρ, t').

1. **If there are no coarse steps for (ρ, t').** Then by Proposition 5.14, the derivation ρ is t'-garbage. Recall that the composition of garbage is again garbage (Proposition 5.4), so $\rho\sigma$ is t'-garbage. Resorting to Proposition 5.14 we obtain that $\rho\sigma \downarrow t' = \epsilon = \rho \downarrow t'$, as required.
2. **If there exists a coarse step for (ρ, t').** Let R_0 be the leftmost coarse step for (ρ, t'). Then since $R_0 \sqsubseteq \rho$ also $R_0 \sqsubseteq \rho\sigma$, so R_0 is a coarse step for $(\rho\sigma, t')$. In particular, since there exists at least one coarse step for $(\rho\sigma, t')$, let S_0 be the leftmost such step. We argue that $R_0 = S_0$. We consider two cases, depending on whether S_0 is coarse for (ρ, t'):
   - 2.1 **If S_0 is coarse for (ρ, t').** Then R_0 and S_0 are both simultaneously coarse for (ρ, t') and for $(\rho\sigma, t')$. Note that R_0 cannot be to the left of S_0, since then S_0 would not be the leftmost coarse step for $(\rho\sigma, t')$. Symmetrically, S_0 cannot be to the left of R_0, since then R_0 would not be the leftmost coarse step for (ρ, t'). Hence $R_0 = S_0$ as claimed.
   - 2.2 **If S_0 is not coarse for (ρ, t').** We argue that this case is impossible. Note that $S_0 \sqsubseteq \rho\sigma$ but it is not the case that $S_0 \sqsubseteq \rho$, so S_0/ρ is not empty. Note also that $S_0/\rho \sqsubseteq \sigma$, so by Corollary 4.20 we have that $(S_0/\rho)/(t'/\rho) \sqsubseteq \sigma/(t'/\rho)$. Moreover, $\sigma/(t'/\rho) = \epsilon$ is empty because σ is (t'/ρ)-garbage. This means that $(S_0/\rho)/(t'/\rho) = \epsilon$. Then we have the following chain of equalities:

$$\begin{array}{rcll} \varnothing & = & \mathsf{names}((S_0/\rho)/(t'/\rho)) & \\ & = & \mathsf{names}((S_0/t')/(\rho/t')) & \text{by Corollary 3.17 and Lemma 4.19} \\ & = & \mathsf{names}(S_0/t') \setminus \mathsf{names}(\rho/t') & \text{by Lemma 3.15} \\ & = & \mathsf{names}(S_0/t') & \end{array}$$

To justify the last step, start by noting that no residual of S_0 is contracted along the derivation ρ. Indeed, if S_0 had a residual, then $\rho = \rho_1 S_1 \rho_2$ where $S_1 \in S_0/\rho$. But recall that S_0 is the leftmost coarse step for $(\rho\sigma, t')$ and $\rho_1 \sqsubseteq \rho\sigma$, so it has at most one residual (Lemma A.30). This means that $S_0/\rho_1 = S_1$, so $S_0/\rho = \varnothing$, which is a contradiction. Given that no residuals of S_0 are contracted along the derivation ρ, we have that the sets $\mathsf{names}(S_0/t')$ and $\mathsf{names}(\rho/t')$ are disjoint which justifies the last step by resorting to Lemma A.27.
According to the chain of equalities above, we have that $\mathsf{names}(S_0/t') = \varnothing$. This means that S_0 is t'-garbage. This in turn contradicts the fact that S_0 is coarse for $(\rho\sigma, t')$, confirming that this case is impossible.

We have just claimed that $R_0 = S_0$. Then:

$$\begin{array}{lcll} \rho\sigma \downarrow t' & = & R_0((\rho\sigma/R_0) \downarrow (t'/R_0)) & \text{by definition of sieving} \\ & = & R_0(((\rho/R_0)(\sigma/(R_0/\rho))) \downarrow (t'/R_0)) & \\ & = & R_0((\rho/R_0) \downarrow (t'/R_0)) & \text{by } \textit{i.h.} \text{ since } \sigma/(R_0/\rho) \text{ is garbage by Proposition 5.4} \\ & = & \rho \downarrow t' & \text{by definition of sieving} \end{array}$$

which concludes the proof.

□

A.20 Proof of Proposition 5.21 — Properties of derivation semi-lattices

To prove Proposition 5.21 we need a few auxiliary lemmas:

Lemma A.34. *Let $\rho \equiv \sigma$. Then $\rho \downarrow t' = \sigma \downarrow t'$.*

Proof. Observe that, given two permutation equivalent derivations ρ and σ, a step R is coarse for (ρ, t') if and only if R is coarse for (σ, t'), since $(R \sqsubseteq \rho) \iff (R/\rho = \varnothing) \iff (R/\sigma = \varnothing) \iff (R \sqsubseteq \sigma)$. Using this observation, the proof is straightforward by induction on the length of $\rho \downarrow t'$. □

Lemma A.35 (Sieving trailing garbage)**.** *Let ρ and σ be composable derivations, and let $t' \ltimes \mathsf{src}(\rho)$. If σ is (t'/ρ)-garbage, then $\rho\sigma \downarrow t' = \rho \downarrow t'$.*

Proof. By induction on the length of $\rho \downarrow t'$. If there are no coarse steps for (ρ, t'), then by Proposition 5.14, the derivation ρ is t'-garbage, so $\rho\sigma$ is t'-garbage by Proposition 5.4. Resorting to Proposition 5.14 we obtain that $\rho\sigma \downarrow t' = \epsilon = \rho \downarrow t'$, as required.

If there exists a coarse step for (ρ, t'), let R_0 be the leftmost such step. Then since $R_0 \sqsubseteq \rho$ also $R_0 \sqsubseteq \rho\sigma$, so R_0 is a coarse step for $(\rho\sigma, t')$. In particular, since there exists at least one coarse step for $(\rho\sigma, t')$, let S_0 be the leftmost such step. We argue that $R_0 = S_0$. We consider two cases, depending on whether S_0 is coarse for (ρ, t'):

1. **If S_0 is coarse for (ρ, t').** Then R_0 and S_0 are both simultaneously coarse for (ρ, t') and for $(\rho\sigma, t')$. Note that R_0 cannot be to the left of S_0, since then S_0 would not be the leftmost coarse step for $(\rho\sigma, t')$. Symmetrically, S_0 cannot be to the left of R_0, since then R_0 would not be the leftmost coarse step for (ρ, t'). Hence $R_0 = S_0$ as claimed.
2. **If S_0 is not coarse for (ρ, t').** We argue that this case is impossible. Note that $S_0 \sqsubseteq \rho\sigma$ but it is not the case that $S_0 \sqsubseteq \rho$, so S_0/ρ is not empty. Note also that $S_0/\rho \sqsubseteq \sigma$, so by Corollary 4.20 we have that $(S_0/\rho)/(t'/\rho) \sqsubseteq \sigma/(t'/\rho)$. Moreover, $\sigma/(t'/\rho) = \epsilon$ is

empty because σ is (t'/ρ)-garbage. This means that $(S_0/\rho)/(t'/\rho) = \epsilon$. Then we have the following chain of equalities:

$$\begin{array}{lcll} \varnothing & = & \mathsf{names}((S_0/\rho)/(t'/\rho)) & \\ & = & \mathsf{names}((S_0/t')/(\rho/t')) & \text{by Corollary 3.17 and Lemma 4.19} \\ & = & \mathsf{names}(S_0/t') \setminus \mathsf{names}(\rho/t') & \text{by Lemma 3.15} \\ & = & \mathsf{names}(S_0/t') & \end{array}$$

To justify the last step, start by noting that no residual of S_0 is contracted along the derivation ρ. Indeed, if S_0 had a residual, then $\rho = \rho_1 S_1 \rho_2$ where $S_1 \in S_0/\rho$. But recall that S_0 is the leftmost coarse step for $(\rho\sigma, t')$ and $\rho_1 \sqsubseteq \rho\sigma$, so it has at most one residual (Lemma A.30). This means that $S_0/\rho_1 = S_1$, so $S_0/\rho = \varnothing$, which is a contradiction. Given that no residuals of S_0 are contracted along the derivation ρ, we have that the sets $\mathsf{names}(S_0/t')$ and $\mathsf{names}(\rho/t')$ are disjoint which justifies the last step.
According to the chain of equalities above, we have that $\mathsf{names}(S_0/t') = \varnothing$. This means that S_0 is t'-garbage. This in turn contradicts the fact that S_0 is coarse for $(\rho\sigma, t')$, confirming that this case is impossible.

We have just claimed that $R_0 = S_0$. Then we conclude as follows:

$$\begin{array}{lcll} \rho\sigma \downarrow t' & = & R_0((\rho\sigma/R_0) \downarrow (t'/R_0)) & \\ & = & R_0(((\rho/R_0)(\sigma/(R_0/\rho))) \downarrow (t'/R_0)) & \\ & = & R_0((\rho/R_0) \downarrow (t'/R_0)) & \text{by } \textit{i.h.}\text{, as } \sigma/(R_0/\rho) \text{ is garbage by Proposition 5.4} \\ & = & \rho \downarrow t' & \end{array}$$

□

Let us check the two items of Proposition 5.21:

1. **The set $\mathbb{F}(t', t)$ forms a finite lattice.** Let us check all the conditions:
   - 1.1 **Partial order.** First let us show that $\trianglelefteq$ is a partial order.
     - 1.1.1 **Reflexivity.** It is immediate that $[\rho] \trianglelefteq [\rho]$ holds since $\rho/\rho = \epsilon$ is garbage.
     - 1.1.2 **Antisymmetry.** Let $[\rho] \trianglelefteq [\sigma] \trianglelefteq [\rho]$. This means that ρ/σ and σ/ρ are garbage. Then:

$$\begin{array}{lcll} \rho & \equiv & \rho \downarrow t' & \text{since } \rho \text{ is garbage-free, by Proposition 5.15} \\ & \equiv & \rho(\sigma/\rho) \downarrow t' & \text{since } \sigma/\rho \text{ is garbage, by Lemma A.35} \\ & \equiv & \sigma(\rho/\sigma) \downarrow t' & \text{since } A(B/A) \equiv B(A/B) \text{ in general, using Lemma A.34} \\ & \equiv & \sigma \downarrow t' & \text{since } \rho/\sigma \text{ is garbage, by Lemma A.35} \\ & \equiv & \sigma & \text{since } \sigma \text{ is garbage-free, by Proposition 5.15} \end{array}$$

       Since $\rho \equiv \sigma$ we conclude that $[\rho] = [\sigma]$, as required.
     - 1.1.3 **Transitivity.** Let $[\rho] \trianglelefteq [\sigma] \trianglelefteq [\tau]$ and let us show that $[\rho] \trianglelefteq [\tau]$. Note that ρ/σ and σ/τ are garbage. By the fact that the projection of garbage is garbage (Proposition 5.4) the derivation $(\rho/\sigma)/(\sigma/\tau)$ is garbage. The composition of garbage is also garbage (Proposition 5.4), so $(\sigma/\tau)((\rho/\sigma)/(\sigma/\tau))$ is garbage. In general the following holds:

$$\begin{array}{lcll} \rho/\tau & \sqsubseteq & (\rho/\tau)((\sigma/\tau)/(\rho/\tau)) & \text{since } A \sqsubseteq AB \text{ in general} \\ & \equiv & (\sigma/\tau)((\rho/\tau)/(\sigma/\tau)) & \text{since } A(B/A) \equiv B(A/B) \text{ in general} \\ & \equiv & (\sigma/\tau)((\rho/\sigma)/(\tau/\sigma)) & \text{since } A(B/A) \equiv B(A/B) \text{ in general} \end{array}$$

So since any prefix of a garbage derivation is garbage (Proposition 5.4) we conclude that ρ/τ is garbage. This means that $[\rho] \trianglelefteq [\tau]$, as required.

1.2 **Finite.** Let us check that $\mathbb{F}(t', t)$ is a finite lattice, *i.e.* that it has a finite number of elements. Recall that $\mathbb{F}(t', t)$ is the set of equivalence classes $[\rho]$ where ρ is t'-garbage-free.

On one hand, recall that the $\lambda^{\#}$-calculus is finitely branching and strongly normalizing (Proposition 2.21). So by König's lemma the set of $\to_{\#}$-derivations starting from t' is bounded in length. More precisely, there is a constant M such that for any $s' \in \mathcal{T}^{\#}$ and any derivation $\rho : t' \to_{\#}^{*} s'$ we have that the length $|\rho|$ is bounded by M.

On the other hand, let $F = \{\rho \mid \rho \text{ is } t'\text{-garbage-free and } \rho \downarrow t' = \rho\}$. Consider the mapping $\varphi : \mathbb{F}(t', t) \to F$ given by $[\rho] \mapsto \rho \downarrow t'$, and note that:

- φ does not depend on the choice of representative, that is, if $[\rho] = [\sigma]$ then $\varphi([\rho]) = \rho \downarrow t' = \sigma \downarrow t' = \varphi([\sigma])$. This is a consequence of Lemma A.34.
- φ is a well-defined function, that is $\varphi([\rho]) \in F$. This is because $\varphi([\rho]) = \rho \downarrow t'$ is t'-garbage-free by Proposition 5.16. Moreover, we have that $\varphi([\rho]) \downarrow t' = ([\rho] \downarrow t') \downarrow t' = [\rho] \downarrow t' = \varphi([\rho])$, which is also a consequence of Lemma A.34, and the fact that $[\rho] \downarrow t' \equiv \rho$ (Proposition 5.16).
- φ is injective. Indeed, suppose that $\varphi([\rho]) = \varphi([\sigma])$, that is, $\rho \downarrow t' = \sigma \downarrow t'$. Then $\rho \equiv \rho \downarrow t' = \sigma \downarrow t' \equiv \sigma$ by Proposition 5.16 since ρ and σ are t'-garbage-free. Hence $[\rho] = [\sigma]$.

Since $\varphi : \mathbb{F}(t', t) \to F$ is injective, it suffices to show that F is finite to conclude that $\mathbb{F}(t', t)$ is finite. Recall that the λ-calculus is finitely branching, so the number of derivations of a certain length is finite. To show that F is finite, it suffices to show that the length of derivations in F is bounded by some constant. Let ρ be a derivation in F. We have that $\rho = \rho \downarrow t'$. By construction of the sieve, none of the steps of $\rho \downarrow t'$ are garbage. That is $\rho = \rho \downarrow t' = R_1 \dots R_n$ where for all i we have that $R_i/(t'/R_1 \dots R_{i-1}) \neq \varnothing$. So we have that the length of ρ/t' is greater than the length of ρ for all $\rho \in F$:

$$|\rho/t'| = \sum_{i=1}^{n} |\underbrace{R/(t'/R_1 \dots R_{i-1})}_{\neq \varnothing}| \geq n = |\rho|$$

As a consequence given any $\to_{\beta}$-derivation $\rho \in F$ we have that $|\rho| \leq |\rho/t'| \leq M$. This concludes the proof that $\mathbb{F}(t', t)$ is finite.

1.3 **Bottom element.** As the bottom element take $\bot_{(\mathbb{F}(t',t))} := [\epsilon]$. Observe that this is well-defined since ϵ is t'-garbage-free. Moreover, let us show that $\bot_{(\mathbb{F}(t',t))}$ is the least element. Let $[\rho]$ be an arbitrary element of $\mathbb{F}(t', t)$ and let us check that $\bot_{(\mathbb{F}(t',t))} \trianglelefteq [\rho]$. This is immediate since, by definition, $\bot_{(\mathbb{F}(t',t))} \trianglelefteq [\rho]$ if and only if ϵ/ρ is garbage. But $\epsilon/\rho = \epsilon$ is trivially garbage.

1.4 **Join.** Let $[\rho], [\sigma]$ be arbitrary elements of $\mathbb{F}(t', t)$, and let us check that $[\rho] \triangledown [\sigma]$ is the join. First observe that $[\rho] \triangledown [\sigma]$ is well-defined *i.e.* that $(\rho \sqcup \sigma) \downarrow t'$ is t'-garbage-free, which is an immediate consequence of Proposition 5.15. Moreover, it is indeed the least upper bound of $\{[\rho], [\sigma]\}$:

1.4.1 **Upper bound.** Let us show that $[\rho] \trianglelefteq [\rho]\triangledown[\sigma]$; the proof for $[\sigma]$ is symmetrical. We must show that $\rho/((\rho \sqcup \sigma) \downarrow t')$ is garbage. Note that $\rho \sqsubseteq \rho \sqcup \sigma$, so in particular $\rho/((\rho \sqcup \sigma) \downarrow t') \sqsubseteq (\rho \sqcup \sigma)/((\rho \sqcup \sigma) \downarrow t')$. Given that any prefix of a garbage derivation is garbage (Proposition 5.4), it suffices to show that $(\rho \sqcup \sigma)/((\rho \sqcup \sigma) \downarrow t')$ is garbage. This is a straightforward consequence of the fact that projecting after a sieve is garbage (Lemma A.32).

1.4.2 **Least upper bound.** Let $[\rho], [\sigma] \trianglelefteq [\tau]$, and let us show that $[\rho]\triangledown[\sigma] \trianglelefteq [\tau]$. We know that ρ/τ and σ/τ are garbage, and we are to show that $((\rho \sqcup \sigma) \downarrow t')/\tau$ is garbage. Note that $(\rho \sqcup \sigma) \downarrow t' \sqsubseteq \rho \sqcup \sigma$ as a consequence of the fact that the sieve is a prefix (Lemma A.28). So in particular $((\rho \sqcup \sigma) \downarrow t')/\tau \sqsubseteq (\rho \sqcup \sigma)/\tau$. Given that any prefix of a garbage derivation is garbage (Proposition 5.4), it suffices to show that $(\rho \sqcup \sigma)/\tau$ is garbage. But $(\rho \sqcup \sigma)/\tau \equiv \rho/\tau \sqcup \sigma/\tau$ so we conclude by the fact that the join of garbage is garbage (Proposition 5.4).

1.5 **Top element.** Since $\mathbb{F}(t', t)$ is finite, its elements are $\{[\tau_1], \ldots, [\tau_n]\}$. It suffices to take $\top := [\tau_1]\triangledown \ldots \triangledown[\tau_n]$ as the top element.

1.6 **Meet.** Let $[\rho], [\sigma] \in \mathbb{F}(t', t)$. Note that the set $L = \{[\tau] \in \mathbb{F}(t', t) \mid [\tau] \trianglelefteq [\rho] \text{ and } [\tau] \trianglelefteq [\sigma]\}$ is finite because, as we have already proved, the set $\mathbb{F}(t', t)$ is itself finite. Then $L = \{[\tau_1], \ldots, [\tau_n]\}$. Take $[\rho] \vartriangle [\sigma] := [\tau_1]\triangledown \ldots \triangledown[\tau_n]$. It is straightforward to show that $[\rho] \vartriangle [\sigma]$ thus defined is the greatest lower bound for $\{[\rho], [\sigma]\}$.

2. **The set $\mathbb{G}(t', t)$ forms an upper semilattice.** The semilattice structure of $\mathbb{G}(t', t)$ is inherited from $\mathbb{D}^\lambda(t)$. More precisely, $[\rho] \sqsubseteq [\sigma]$ is declared to hold if $\rho \sqsubseteq \sigma$, the bottom element is $[\epsilon]$, and the join is $[\rho] \sqcup [\sigma] = [\rho \sqcup \sigma]$. Let us check that this is an upper semilattice:

2.1 **Partial order.** The relation $(\sqsubseteq)$ is a partial order on $\mathbb{G}(t', t)$ because it is already a partial order in $\mathbb{D}^\lambda(t)$.

2.2 **Bottom element.** It suffices to note that the empty derivation $\epsilon : t \to_\beta{}^* t$ is t'-garbage, so $[\epsilon] \in \mathbb{G}(t', t)$ is the least element.

2.3 **Join.** By Proposition 5.4 we know that if ρ and σ are t'-garbage then $\rho \sqcup \sigma$ is t'-garbage, so $[\rho] \sqcup [\sigma]$ is indeed the least upper bound of $\{[\rho], [\sigma]\}$.

A.21 Proof of Theorem 5.22 — Factorization

We need a few auxiliary lemmas:

Lemma A.36 (Garbage-free/garbage decomposition)**.** *Let $\rho : t \twoheadrightarrow_\beta s$ and $t' \ltimes t$. Then there exist ρ_1, ρ_2 such that:*

1. *$\rho \equiv \rho_1\rho_2$*
2. *ρ_1 is t'-garbage-free,*
3. *ρ_2 is t'-garbage.*

Moreover, ρ_1 and ρ_2 are unique modulo permutation equivalence, and we have that $\rho_1 \equiv \rho \downarrow t'$ and $\rho_2 \equiv \rho/(\rho \downarrow t')$.

Proof. Let us prove that said decomposition exists and that it is unique:

- **Existence.** Take $\rho_1 := \rho \downarrow t'$ and $\rho_2 := \rho/(\rho \downarrow t')$. Then we may check the three conditions in the statement:
  1. Recall that $\rho \downarrow t' \sqsubseteq \rho$ holds by Lemma A.28, so:
  $$\begin{array}{rcll} \rho_1\rho_2 & = & (\rho \downarrow t')(\rho/(\rho \downarrow t')) & \\ & \equiv & \rho((\rho \downarrow t')/\rho) & \text{since } A(B/A) \equiv B(A/B) \text{ holds in general} \\ & \equiv & \rho & \text{since } \rho \downarrow t' \sqsubseteq \rho \end{array}$$
  as required.
  2. The derivation $\rho_1 = \rho \downarrow t'$ is t'-garbage-free. This is as an immediate consequence of Proposition 5.15, namely the result of sieving is always garbage-free.
  3. The derivation $\rho_2 = \rho/(\rho \downarrow t')$ is garbage. This is an immediate consequence of Lemma A.32, namely projection after a sieve is always garbage.
- **Uniqueness, modulo permutation equivalence.** Let $\rho \equiv \sigma_1, \sigma_2$ where σ_1 is t'-garbage-free, and σ_2 is t'-garbage. Then we argue that $\sigma_1 \equiv \rho_1$ and $\sigma_2 \equiv \rho_2$.

  Since $\rho_1\rho_2 \equiv \rho \equiv \sigma_1\sigma_2$ and sieving is compatible with permutation equivalence (Lemma A.34) we have that $\rho_1\rho_2 \downarrow t' \equiv \sigma_1\sigma_2 \downarrow t'$. By Lemma A.35, we know that trailing garbage does not affect sieving, hence $\rho_1 \downarrow t' \equiv \sigma_1 \downarrow t'$. Moreover, ρ_1 and σ_1 are garbage-free, which by Proposition 5.15 means that $\rho_1 \equiv \sigma_1$. This means that ρ_1 is unique, modulo permutation equivalence. Finally, since $\rho_1\rho_2 \equiv \sigma_1\sigma_2$ and $\rho_1 \equiv \sigma_1$, we have that $\rho_1\rho_2/\rho_1 \equiv \sigma_1\sigma_2/\sigma_1$, that is $\rho_2 \equiv \sigma_2$. This means that ρ_2 is unique, modulo permutation equivalence.

□

Lemma A.37. *Let* $t' \ltimes \mathsf{src}(\rho) = \mathsf{src}(\sigma)$. *Then* $[(\rho \sqcup \sigma) \downarrow t'] = [\rho \downarrow t'] \triangledown [\sigma \downarrow t']$.

Proof. Let:
$$\begin{array}{rcl} \alpha & := & \rho/(\rho \downarrow t') \\ \beta & := & \sigma/(\sigma \downarrow t') \\ \gamma & := & (\alpha/((\sigma \downarrow t')/(\rho \downarrow t'))) \sqcup (\beta/((\rho \downarrow t')/(\sigma \downarrow t'))) \end{array}$$
Note that α and β are garbage by Lemma A.32 and hence γ is also garbage, as a consequence of the facts that the join of garbage is garbage and that the projection of garbage is garbage (Proposition 5.4). Remark that, in general, $AB \sqcup CD \equiv (A \sqcup C)(B/(C/A) \sqcup D/(A/C))$. Then the statement of this lemma is a consequence of the following chain of equalities:
$$\begin{array}{rcll} [(\rho \sqcup \sigma) \downarrow t'] & = & [((\rho \downarrow t')\alpha \sqcup (\sigma \downarrow t')\beta) \downarrow t'] & \text{by Lemma A.36} \\ & = & [((\rho \downarrow t') \sqcup (\sigma \downarrow t'))\gamma \downarrow t'] & \text{by the previous remark} \\ & = & [((\rho \downarrow t') \sqcup (\sigma \downarrow t')) \downarrow t'] & \text{by Lemma A.35} \\ & = & [\rho \downarrow t'] \triangledown [\sigma \downarrow t'] & \text{by definition of } \triangledown \end{array}$$

□

To prove Theorem 5.22, let us check that $\int_{\mathcal{F}} \mathcal{G}$ is well-defined, that it is an upper semilattice, and finally that $\mathbb{D}^\lambda(t) \simeq \int_{\mathcal{F}} \mathcal{G}$ are isomorphic as upper semilattices.

Proof. 1. **The Grothendieck construction $\int_{\mathcal{F}} \mathcal{G}$ is well-defined.** This amounts to checking that $\mathcal{G}$ is indeed a lax 2-functor:

1.1 **The mapping $\mathcal{G}$ is well-defined on objects.** Note that $\mathcal{G}([\rho]) = \mathbb{G}(t'/\rho, \mathsf{tgt}(\rho))$, which is a poset. Moreover, the choice of the representative ρ of the equivalence class $[\rho]$ does not matter, since if ρ and σ are permutation equivalent derivations, then $t'/\rho = t'/\sigma$ (by Proposition 4.17) and $\mathsf{tgt}(\rho) = \mathsf{tgt}(\sigma)$.

1.2 **The mapping $\mathcal{G}$ is well-defined on morphisms.** Let us check that given $[\rho], [\sigma] \in \mathcal{F}$ such that $[\rho] \trianglelefteq [\sigma]$ then $\mathcal{G}([\rho] \hookrightarrow_{\mathcal{F}} [\sigma]) : \mathcal{G}([\rho]) \to \mathcal{G}([\sigma])$ is a morphism of posets, *i.e.* a monotonic function.
First, we can see that it is well-defined, since if $[\alpha] \in \mathcal{G}([\rho])$ then the image $\mathcal{G}([\rho])([\alpha] \hookrightarrow_{\mathcal{F}} [\sigma]) = [\rho\alpha/\sigma]$ is an element of $\mathcal{G}([\sigma])$, since $\rho\alpha/\sigma = (\rho/\sigma)(\alpha/(\sigma/\rho))$ is garbage, as it is the composition of garbage derivations (Proposition 5.4): the derivation ρ/σ is garbage since $[\rho] \trianglelefteq [\sigma]$ (by definition), and the derivation $\alpha/(\sigma/\rho)$ is garbage since α is garbage (Proposition 5.4). Moreover, the choice of representative does not matter, since if $\rho_1 \equiv \rho_2$ and $\sigma_1 \equiv \sigma_2$ and $\alpha_1 \equiv \alpha_2$ then $\rho_1\alpha_1/\sigma_1 \equiv \rho_2\alpha_2/\sigma_2$.
We are left to verify that $\mathcal{G}([\rho] \hookrightarrow_{\mathcal{F}} [\sigma])$ is monotonic. Let $[\alpha], [\beta] \in \mathcal{G}([\rho])$ such that $[\alpha] \sqsubseteq [\beta]$, and let us show that $\mathcal{G}([\rho] \hookrightarrow_{\mathcal{F}} [\sigma])([\alpha]) \sqsubseteq \mathcal{G}([\rho] \hookrightarrow_{\mathcal{F}} [\sigma])([\beta])$. Indeed, $\alpha \sqsubseteq \beta$, so $\rho\alpha/\sigma = (\rho/\sigma)(\alpha/(\sigma/\rho)) \sqsubseteq (\rho/\sigma)(\beta/(\sigma/\rho)) = \rho\beta/\sigma$.

1.3 **Identity.** Let $[\rho] \in \mathcal{F}$. Let us check that $\mathcal{G}([\rho] \hookrightarrow_{\mathcal{F}} [\rho]) = \mathrm{id}_{\mathcal{G}([\rho])}$ is the identity morphism. Indeed, if $[\alpha] \in \mathcal{G}([\rho])$, then $\mathcal{G}([\rho] \hookrightarrow_{\mathcal{F}} [\rho])([\alpha]) = [\rho\alpha/\rho] = [\alpha]$.

1.4 **Composition.** Let $[\rho], [\sigma], [\tau] \in \mathcal{F}$ such that $[\rho] \trianglelefteq [\sigma] \trianglelefteq [\tau]$. Let us check that there is a 2-cell $\mathcal{G}(([\sigma] \hookrightarrow_{\mathcal{F}} [\tau]) \circ ([\rho] \hookrightarrow_{\mathcal{F}} [\sigma])) \sqsubseteq \mathcal{G}([\sigma] \hookrightarrow_{\mathcal{F}} [\tau]) \circ \mathcal{G}([\rho] \hookrightarrow_{\mathcal{F}} [\sigma])$. Note that $([\sigma] \hookrightarrow_{\mathcal{F}} [\tau]) \circ ([\rho] \hookrightarrow_{\mathcal{F}} [\sigma]) : [\rho] \trianglelefteq [\tau]$ is a morphism in the upper semilattice $\mathcal{F}$ seen as a category. Moreover, since it is a semilattice, there a unique morphism $[\rho] \trianglelefteq [\tau]$, namely $[\rho] \hookrightarrow_{\mathcal{F}} [\tau]$, so we have that $([\sigma] \hookrightarrow_{\mathcal{F}} [\tau]) \circ ([\rho] \hookrightarrow_{\mathcal{F}} [\sigma]) = [\rho] \hookrightarrow_{\mathcal{F}} [\tau]$. Now if $[\alpha] \in \mathcal{G}[\rho]$, then:

$$\begin{array}{lcll}
\mathcal{G}(([\sigma] \hookrightarrow_{\mathcal{F}} [\tau]) \circ ([\rho] \hookrightarrow_{\mathcal{F}} [\sigma]))([\alpha]) & = & \mathcal{G}([\rho] \hookrightarrow_{\mathcal{F}} [\tau])([\alpha]) & \\
& = & \rho\alpha/\tau & \\
& \sqsubseteq & (\rho\alpha/\tau)((\sigma/\rho\alpha)/(\tau/\rho\alpha)) & \\
& = & \rho\alpha(\sigma/\rho\alpha)/\tau & \\
& \equiv & \sigma(\rho\alpha/\sigma)/\tau & \text{since } A(B/A) \equiv B(A/B) \\
& = & \mathcal{G}([\sigma] \hookrightarrow_{\mathcal{F}} [\tau])(\rho\alpha/\sigma) & \\
& = & (\mathcal{G}([\sigma] \hookrightarrow_{\mathcal{F}} [\tau]) \circ \mathcal{G}([\rho] \hookrightarrow_{\mathcal{F}} [\sigma]))([\alpha]) &
\end{array}$$

so $\mathcal{G}(([\sigma] \hookrightarrow_{\mathcal{F}} [\tau]) \circ ([\rho] \hookrightarrow_{\mathcal{F}} [\sigma])) \sqsubseteq \mathcal{G}([\sigma] \hookrightarrow_{\mathcal{F}} [\tau]) \circ \mathcal{G}([\rho] \hookrightarrow_{\mathcal{F}} [\sigma])$ as required.

2. **The Grothendieck construction $\int_{\mathcal{F}} \mathcal{G}$ is an upper semilattice.**

2.1 **Partial order.** Recall that $\int_{\mathcal{F}} \mathcal{G}$ is always poset with the order given by $(a, b) \leq (a', b')$ if and only if $a \trianglelefteq a'$ and $\mathcal{G}(a \hookrightarrow_{\mathcal{F}} a')(b) \sqsubseteq b'$.

2.2 **Bottom element.** We argue that $(\bot_{\mathcal{F}}, \bot_{\mathcal{G}(\bot_{\mathcal{F}})})$ is the bottom element. Let $([\rho], [\sigma]) \in \int_{\mathcal{F}} \mathcal{G}$. Then clearly $\bot_{\mathcal{F}} \trianglelefteq [\rho]$. Moreover, $\mathcal{G}([\bot_{\mathcal{F}}] \hookrightarrow_{\mathcal{F}} [\rho])(\bot_{\mathcal{G}}) = [\epsilon/\rho] = [\epsilon] \sqsubseteq [\sigma]$.

2.3 **Join.** Let us show that $(a, b) \vee (a', b') = (a \triangledown a', \mathcal{G}(a \hookrightarrow_{\mathcal{F}} (a \triangledown a'))(b) \sqcup \mathcal{G}(a' \hookrightarrow_{\mathcal{F}} (a \triangledown a'))(b'))$. is the least upper bound of $\{(a, b), (a', b')\}$.

2.3.1 **Upper bound.** Let us show that $(a, b) \leq (a, b) \vee (a', b')$. Recall that $(a, b) \vee (a', b') = (a \triangledown a, \mathcal{G}(a \hookrightarrow_{\mathcal{F}} (a \vee a'))(b) \sqcup \mathcal{G}(a' \hookrightarrow_{\mathcal{F}} (a \triangledown a'))(b'))$. First, we have $a \trianglelefteq a \triangledown a'$. Moreover, $\mathcal{G}(a \hookrightarrow_{\mathcal{F}} (a \triangledown a'))(b) \sqsubseteq \mathcal{G}(a \hookrightarrow_{\mathcal{F}} (a \vee a'))(b) \sqcup \mathcal{G}(a' \hookrightarrow_{\mathcal{F}} (a \triangledown a'))(b')$, as required.

2.3.2 **Least upper bound.** Let $(a, b) = ([\rho], [\sigma])$ and $(a', b') = ([\rho'], [\sigma'])$. Moreover, let $([\rho''], [\sigma''])$ be an upper bound, *i.e.* an element such that $([\rho], [\sigma]) \leq ([\rho''], [\sigma''])$ and $([\rho'], [\sigma']) \leq ([\rho''], [\sigma''])$. Let us show that $([\rho], [\sigma]) \vee ([\rho'], [\sigma']) \leq ([\rho''], [\sigma''])$.
First note that $[\rho] \trianglelefteq [\rho'']$ and $[\rho'] \trianglelefteq [\rho'']$ so $[\rho] \triangledown [\rho'] \trianglelefteq [\rho'']$.
Moreover, we know that:

$$[\rho\sigma/\rho''] = \mathcal{G}([\rho] \hookrightarrow [\rho''])([\sigma]) \sqsubseteq [\sigma''] \text{ and } [\rho'\sigma'/\rho''] = \mathcal{G}([\rho'] \hookrightarrow [\rho''])([\sigma']) \sqsubseteq [\sigma'']$$

Let $\alpha := (\rho \sqcup \rho') \downarrow t'$. First we claim that $\alpha \sqsubseteq \rho\sigma \sqcup \rho'\sigma'$. Indeed, $\alpha = (\rho \sqcup \rho') \downarrow t' \sqsubseteq \rho \sqcup \rho'$ by Lemma A.28, and it is easy to check that $\rho \sqcup \rho' \sqsubseteq \rho\sigma \sqcup \rho'\sigma'$. What

we have to check is the following inequality:

$$\mathcal{G}([\alpha] \hookrightarrow [\rho''])(\mathcal{G}([\rho] \hookrightarrow [\alpha])([\sigma]) \sqcup \mathcal{G}([\rho'] \hookrightarrow [\alpha])([\sigma'])) \sqsubseteq [\sigma'']$$

Indeed:

$$\begin{array}{lll} & \mathcal{G}([\alpha] \hookrightarrow [\rho''])(\mathcal{G}([\rho] \hookrightarrow [\alpha])([\sigma]) \sqcup \mathcal{G}([\rho'] \hookrightarrow [\alpha])([\sigma'])) & \\ = & [\alpha((\rho\sigma/\alpha) \sqcup (\rho'\sigma'/\alpha))/\rho''] & \\ = & [\alpha((\rho\sigma \sqcup \rho'\sigma')/\alpha)/\rho''] & \text{since } A/C \sqcup B/C \equiv (A \sqcup B)/C \\ = & [(\rho\sigma \sqcup \rho'\sigma')(\alpha/(\rho\sigma \sqcup \rho'\sigma'))/\rho''] & \text{since } A(B/A) \equiv B(A/B) \\ = & [(\rho\sigma \sqcup \rho'\sigma')/\rho''] & \text{since } \alpha \sqsubseteq \rho\sigma \sqcup \rho'\sigma' \\ = & [\rho\sigma/\rho'' \sqcup \rho'\sigma'/\rho''] & \text{since } A/C \sqcup B/C \equiv (A \sqcup B)/C \\ \sqsubseteq & [\sigma''] & \text{since } [\rho\sigma/\rho''] \sqsubseteq [\sigma''], [\rho'\sigma'/\rho''] \sqsubseteq [\sigma''] \end{array}$$

3. **There is an isomorphism $\mathbb{D}^\lambda(t) \simeq \int_{\mathcal{F}} \mathcal{G}$ of upper semilattices.** As stated, the isomorphism is given by:

$$\begin{array}{rcccl} \varphi & : & \mathbb{D}^\lambda(t) & \to & \int_{\mathcal{F}} \mathcal{G} \\ & & [\rho] & \mapsto & ([\rho \downarrow t'], [\rho/(\rho \downarrow t')]) \end{array}$$

$$\begin{array}{rcccl} \psi & : & \int_{\mathcal{F}} \mathcal{G} & \to & \mathbb{D}^\lambda(t) \\ & & ([\rho], [\sigma]) & \mapsto & [\rho\sigma] \end{array}$$

Note that φ and ψ are well-defined mappings, since their value does not depend on the choice of representative, due, in particular, to the fact that sieving is compatible with permutation equivalence (Lemma A.34). Let us check that φ and ψ are morphisms of upper semilatices, and that they are mutual inverses:

3.1 **φ is a morphism of upper semilattices.** Let us check the three conditions:

3.1.1 **Monotonic.** Let $[\rho] \sqsubseteq [\sigma]$ in $\mathbb{D}^\lambda(t)$, and let us show that the following inequality holds:

$$\varphi([\rho]) = ([\rho \downarrow t'], [\rho/(\rho \downarrow t')]) \leq ([\sigma \downarrow t'], [\sigma/(\sigma \downarrow t')]) = \varphi([\sigma])$$

We check the two conditions (by definition of $\int_{\mathcal{F}} \mathcal{G}$):

3.1.1.1 On the first hand, $[\rho \downarrow t'] \trianglelefteq [\sigma \downarrow t']$ since

$$\begin{array}{lll} (\rho \downarrow t')/(\sigma \downarrow t') & \sqsubseteq & \rho/(\sigma \downarrow t') \quad \text{since } \rho \downarrow t' \sqsubseteq \rho \text{ by Lemma A.28} \\ & \sqsubseteq & \sigma/(\sigma \downarrow t') \quad \text{since } \rho \sqsubseteq \sigma \text{ by hypothesis} \end{array}$$

Note that this is garbage by Lemma A.32. So by Proposition 5.4, $(\rho \downarrow t')/(\sigma \downarrow t')$ is also garbage, as required.

3.1.1.2 On the other hand, let us show that $\mathcal{G}([\rho \downarrow t'] \hookrightarrow_{\mathcal{F}} [\sigma \downarrow t'])([\rho/(\rho \downarrow t')]) \sqsubseteq \sigma/(\sigma \downarrow t')$. In fact:

$$\begin{array}{lll} \mathcal{G}([\rho \downarrow t'] \hookrightarrow_{\mathcal{F}} [\sigma \downarrow t'])([\rho/(\rho \downarrow t')]) & = & [(\rho \downarrow t')(\rho/(\rho \downarrow t'))/(\sigma \downarrow t')] \quad \text{by definition} \\ & = & [\rho/(\sigma \downarrow t')] \quad \text{by Lemma A.36} \\ & \sqsubseteq & [\sigma/(\sigma \downarrow t')] \quad \text{since } \rho \sqsubseteq \sigma \end{array}$$

3.1.2 **Preserves bottom.** By definition: $\varphi(\bot_{\mathbb{D}^\lambda(t)}) = ([\epsilon \downarrow t'], [\epsilon/(\epsilon \downarrow t')]) = ([\epsilon], [\epsilon]) = (\bot_{\mathcal{F}}, \bot_{\mathcal{G}(\bot_{\mathcal{F}})})$.

3.1.3 **Preserves joins.** Let $[\rho], [\sigma] \in \mathbb{D}^\lambda(t)$, and let us show that $\varphi([\rho] \sqcup [\sigma]) = \varphi([\sigma]) \vee \varphi([\sigma])$. Indeed, note that:

$$\varphi([\rho] \sqcup [\sigma]) = (\alpha, \beta)$$

where

$$\begin{array}{rcl} \alpha & = & [(\rho \sqcup \sigma) \downarrow t'] \\ \beta & = & [(\rho \sqcup \sigma)/((\rho \sqcup \sigma) \downarrow t')] \end{array}$$

and

$$\varphi([\rho]) \vee \varphi([\sigma]) = ([\rho \downarrow t'], [\rho/(\rho \downarrow t')]) \vee ([\sigma \downarrow t'], [\sigma/(\sigma \downarrow t')]) = (\alpha', \beta')$$

where

$$\begin{array}{rcl} \alpha' & = & [\rho \downarrow t'] \triangledown [\sigma \downarrow t'] \\ \beta' & = & \mathcal{G}([\rho \downarrow t'] \hookrightarrow_{\mathcal{F}} \alpha)([\rho/(\rho \downarrow t')]) \sqcup \mathcal{G}([\sigma \downarrow t'] \hookrightarrow_{\mathcal{F}} \alpha)([\sigma/(\sigma \downarrow t')]) \end{array}$$

It suffices to show that $\alpha = \alpha'$ and $\beta = \beta'$. Let us show each separately:

3.1.3.1 **Proof of $\alpha = \alpha'$.** The equality $\alpha = [(\rho \sqcup \sigma) \downarrow t'] = [\rho \downarrow t'] \triangledown [\sigma \downarrow t'] = \alpha'$ is an immediate consequence of Lemma A.37.

3.1.3.2 **Proof of $\beta = \beta'$.** Note that:

$$\begin{array}{rcll} \beta' & = & \mathcal{G}([\rho \downarrow t'] \hookrightarrow_{\mathcal{F}} \alpha')([\rho/(\rho \downarrow t')]) \sqcup \mathcal{G}([\sigma \downarrow t'] \hookrightarrow_{\mathcal{F}} \alpha')([\sigma/(\sigma \downarrow t')]) & \\ & = & [(\rho \downarrow t')(\rho/(\rho \downarrow t'))/\alpha' \sqcup (\sigma \downarrow t')(\sigma/(\sigma \downarrow t'))/\alpha'] & \\ & = & [\rho/\alpha' \sqcup \sigma/\alpha'] & \text{by Lemma A.36} \\ & = & [(\rho \sqcup \sigma)/\alpha'] & \text{since } A/C \sqcup B/C \equiv (A \sqcup B)/C \\ & = & [(\rho \sqcup \sigma)/((\rho \sqcup \sigma) \downarrow t')] & \text{since } \alpha' = \alpha = (\rho \sqcup \sigma) \downarrow t' \\ & = & \beta & \end{array}$$

as required.

3.2 **ψ is a morphism of upper semilattices.** Let us check the three conditions:

3.2.1 **Monotonic.** Let $([\rho_1], [\sigma_1]) \leq ([\rho_2], [\sigma_2])$ in $\int_{\mathcal{F}} \mathcal{G}$ and let us show that $\psi([\rho_1], [\sigma_1]) \sqsubseteq \psi([\rho_2], [\sigma_2])$ in $\mathbb{D}^\lambda(t)$. Indeed, we know that $\mathcal{G}([\rho_1] \hookrightarrow_{\mathcal{F}} [\rho_2])([\sigma_1]) \sqsubseteq [\sigma_2]$, that is to say $\rho_1\sigma_1/\rho_2 \sqsubseteq \sigma_2$. Then:

$$\rho_1\sigma_1/\rho_2\sigma_2 = (\rho_1\sigma_1/\rho_2)/\sigma_2 = \epsilon$$

which means that $\rho_1\sigma_1 \sqsubseteq \rho_2\sigma_2$. This immediately implies that $\psi([\rho_1], [\sigma_1]) \sqsubseteq \psi([\rho_2], [\sigma_2])$.

3.2.2 **Preserves bottom.** Recall that the bottom element $\bot_{(\int_{\mathcal{F}} \mathcal{G})}$ is defined as $(\bot_{\mathcal{F}}, \bot_{\mathcal{G}(\bot_{\mathcal{F}})})$, that is $([\epsilon], [\epsilon])$. Therefore $\psi(\bot_{(\int_{\mathcal{F}} \mathcal{G})}) = [\epsilon] = \bot_{\mathbb{D}^\lambda(t)}$.

3.2.3 **Preserves joins.** Let $([\rho_1], [\sigma_1])$ and $([\rho_2], [\sigma_2])$ be elements of $\int_{\mathcal{F}} \mathcal{G}$, and let us show that $\psi(([\rho_1], [\sigma_1]) \vee ([\rho_2], [\sigma_2])) = \psi([\rho_1], [\sigma_1]) \sqcup \psi([\rho_2], [\sigma_2]))$.
Let:

$$\alpha \quad := \quad (\rho_1 \sqcup \rho_2) \downarrow t'$$

First we claim that $\alpha \sqsubseteq \rho_1\sigma_1 \sqcup \rho_2\sigma_2$. This is because by Lemma A.28 we know that $\alpha = (\rho_1 \sqcup \rho_2) \downarrow t' \sqsubseteq \rho_1 \sqcup \rho_2$. Moreover, it is easy to check that $\rho_1 \sqcup \rho_2 \sqsubseteq$

$\rho_1\sigma_1 \sqcup \rho_2\sigma_2$. Using this fact we have:

$$\begin{array}{rcl}
\psi(([\rho_1],[\sigma_1]) \vee ([\rho_2],[\sigma_2])) & = & \psi([\alpha], \mathcal{G}([\rho_1] \hookrightarrow_{\mathcal{F}} [\alpha])([\sigma_1]) \sqcup \mathcal{G}([\rho_2] \hookrightarrow_{\mathcal{F}} [\alpha])([\sigma_2])) \\
& = & \psi([\alpha], [(\rho_1\sigma_1/\alpha) \sqcup (\rho_2\sigma_2/\alpha)]) \\
& = & \psi([\alpha], [(\rho_1\sigma_1 \sqcup \rho_2\sigma_2)/\alpha]) \\
& & \text{since } A/C \sqcup B/C \sqsubseteq (A \sqcup B)/C \\
& = & [\alpha((\rho_1\sigma_1 \sqcup \rho_2\sigma_2)/\alpha)] \\
& = & [(\rho_1\sigma_1 \sqcup \rho_2\sigma_2)(\alpha/(\rho_1\sigma_1 \sqcup \rho_2\sigma_2))] \\
& = & [\rho_1\sigma_1 \sqcup \rho_2\sigma_2] \\
& & \text{since } \alpha \sqsubseteq \rho_1\sigma_1 \sqcup \rho_2\sigma_2, \text{ so } \alpha/(\rho_1\sigma_1 \sqcup \rho_2\sigma_2) = \epsilon \\
& = & \psi([\rho_1],[\sigma_1]) \sqcup \psi([\rho_2],[\sigma_2]))
\end{array}$$

as required.

3.3 **Left inverse:** $\psi \circ \varphi = \mathsf{id}$**.** Let $[\rho] \in \mathbb{D}^{\lambda}(t)$. Then by Lemma A.36:

$$\psi(\varphi([\rho])) = \psi([\rho \downarrow t'], [\rho/(\rho \downarrow t')]) = [(\rho \downarrow t')(\rho/(\rho \downarrow t'))] = [\rho]$$

3.4 **Right inverse:** $\varphi \circ \psi = \mathsf{id}$**.** Let $([\rho],[\sigma]) \in \int_{\mathcal{F}} \mathcal{G}$. Note that ρ is t'-garbage-free and σ is t'-garbage, so by Lemma A.35 and Proposition 5.15 we know that $\rho\sigma \downarrow t' = \rho \downarrow t' \equiv \rho$. Hence:

$$\varphi(\psi([\rho],[\sigma])) = \varphi([\rho\sigma]) = ([\rho\sigma \downarrow t'], [\rho\sigma/(\rho\sigma \downarrow t')]) = ([\rho],[\rho\sigma/\rho]) = ([\rho],[\sigma])$$

□

Bibliography

[ABKL14] Beniamino Accattoli, Eduardo Bonelli, Delia Kesner, and Carlos Lombardi. A nonstandard standardization theorem. In *POPL '14, San Diego, CA, USA, January 20-21, 2014*, pages 659–670, 2014.

[AL13] Andrea Asperti and Jean-Jacques Lévy. The cost of usage in the lambda-calculus. In *28th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2013, New Orleans, LA, USA, June 25-28, 2013*, pages 293–300, 2013.

[Bar84] Henk Barendregt. *The Lambda Calculus: Its Syntax and Semantics*, volume 103. Elsevier, 1984.

[BKDR14] Antonio Bucciarelli, Delia Kesner, and Simona Ronchi Della Rocca. The inhabitation problem for non-idempotent intersection types. In *IFIP International Conference on Theoretical Computer Science*, pages 341–354. Springer, 2014.

[BKV17] Antonio Bucciarelli, Delia Kesner, and Daniel Ventura. Non-idempotent intersection types for the lambda-calculus. *Logic Journal of the IGPL*, 25(4):431–464, 2017.

[BL13] Alexis Bernadet and Stéphane Jean Lengrand. Non-idempotent intersection types and strong normalisation. *arXiv preprint arXiv:1310.1622*, 2013.

[BMPR16] Flavien Breuvart, Giulio Manzonetto, Andrew Polonsky, and Domenico Ruoppolo. New results on morris's observational theory: The benefits of separating the inseparable. In *1st International Conference on Formal Structures for Computation and Deduction (FSCD)*, 2016.

[Bou93] Gérard Boudol. The lambda-calculus with multiplicities. In *CONCUR'93*, pages 1–6. Springer, 1993.

[Car07] Daniel de Carvalho. *Sémantiques de la logique linéaire et temps de calcul*. PhD thesis, Ecole Doctorale Physique et Sciences de la Matière (Marseille), 2007.

[CD78] Mario Coppo and Mariangiola Dezani-Ciancaglini. A new type assignment for lambda-terms. *Arch. Math. Log.*, 19(1):139–156, 1978.

[CF58] H.B. Curry and R. Feys. *Combinatory Logic*. Number v. 1 in Combinatory Logic. North-Holland Publishing Company, 1958.

[CR36] Alonzo Church and J Barkley Rosser. Some properties of conversion. *Transactions of the American Mathematical Society*, 39(3):472–482, 1936.

[DCGd98] Mariangiola Dezani-Ciancaglini, Elio Giovannetti, and Ugo de'Liguoro. *Intersection Types, λ-models, and Böhm Trees*, volume Volume 2 of *MSJ Memoirs*, pages 45–97. The Mathematical Society of Japan, Tokyo, Japan, 1998.

[DJK91] Nachum Dershowitz, Jean-Pierre Jouannaud, and Jan Willem Klop. Open problems in rewriting. In *International Conference on Rewriting Techniques and Applications*, pages 445–456. Springer, 1991.

[Ehr12] Thomas Ehrhard. Collapsing non-idempotent intersection types. In *LIPIcs-Leibniz International Proceedings in Informatics*, volume 16. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2012.

[ER03] Thomas Ehrhard and Laurent Regnier. The differential lambda-calculus. *Theoretical Computer Science*, 309(1):1–41, 2003.

[Gar94] Philippa Gardner. Discovering needed reductions using type theory. In *Theoretical Aspects of Computer Software*, pages 555–574. Springer, 1994.

[Gir87] Jean-Yves Girard. Linear logic. *Theoretical computer science*, 50(1):1–101, 1987.

[JHM69] Jr James Hiram Morris. *Lambda-Calculus Models of Programming Languages*. PhD thesis, Massachusetts Institute of Technology, 1969.

[Kes16] Delia Kesner. Reasoning about call-by-need by means of types. In *International Conference on Foundations of Software Science and Computation Structures*, pages 424–441. Springer, 2016. LNCS 9634:424-441

[KL07] Delia Kesner and Stéphane Lengrand. Resource operators for λ-calculus. *Information and Computation*, 205(4):419–473, 2007.

[KR] Delia Kesner and Fabien Renaud. The prismoid of resources. Springer. MFCS 2009 (LNCS 5733:464-476)

[KRV18] Delia Kesner, Alejandro Ríos, and Andrés Viso. Call-by-need, neededness and all that. In *21st International Conference on Foundations of Software Science and Computation Structures (FoSSaCS)*, 2018. LNCS 10803:241-257

[Lan94] Cosimo Laneve. Distributive evaluations of λ-calculus. *Fundamenta Informaticae*, 20(4):333–352, 1994.

[Lév78] Jean-Jacques Lévy. *Réductions correctes et optimales dans le lambda-calcul*. PhD thesis, Université de Paris 7, 1978.

[Lev15] Jean-Jacques Levy. Redexes are stable in the λ-calculus. 27:1–13, 07 2015.

[Mel96] Paul-André Melliès. *Description Abstraite des Systèmes de Réécriture*. PhD thesis, Université Paris 7, december 1996.

[Mel97] Paul-André Melliès. A factorisation theorem in rewriting theory. In *Category Theory and Computer Science, 7th International Conference, CTCS '97, Santa Margherita Ligure, Italy, September 4-6, 1997, Proceedings*, pages 49–68, 1997.

[Mel00] Paul-André Melliès. Axiomatic rewriting theory II: the $\lambda\sigma$-calculus enjoys finite normalisation cones. *J. Log. Comput.*, 10(3):461–487, 2000.

[Mel02a] Paul-André Melliès. Axiomatic rewriting theory VI residual theory revisited. In Sophie Tison, editor, *Rewriting Techniques and Applications, 13th International Conference, RTA 2002, Copenhagen, Denmark, July 22-24, 2002, Proceedings*, volume 2378 of *Lecture Notes in Computer Science*, pages 24–50. Springer, 2002.

[Mel02b] Paul-André Mellies. Axiomatic rewriting theory vi: Residual theory revisited. In *Rewriting techniques and applications*, pages 5–11. Springer, 2002.

[Mel05] Paul-André Melliès. Axiomatic rewriting theory I: A diagrammatic standardization theorem. In *Processes, Terms and Cycles: Steps on the Road to Infinity, Essays Dedicated to Jan Willem Klop, on the Occasion of His 60th Birthday*, pages 554–638, 2005.

[Str72] Ross Street. *Two constructions on Lax functors*. Cahiers de Topologie et Géométrie Différentielle Catégoriques. 1972.

[Ter03] Terese. *Term Rewriting Systems*, volume 55 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 2003.

[Via17] Pierre Vial. *Non-Idempotent Typing Operators, Beyond the Lambda-Calculus*. PhD thesis, Université Paris 7, december 2017.

[Zil84] Marisa Venturini Zilli. Reduction graphs in the lambda calculus. *Theor. Comput. Sci.*, 29:251–275, 1984.