A Deep Quantitative Type System

Giulio Guerrieri
University of Bath, Department of Computer Science, Bath, United Kingdom.

Willem B. Heijltjes
University of Bath, Department of Computer Science, Bath, United Kingdom.
http://willem.heijltj.es/

Joseph W.N. Paulus
Rijksuniversiteit Groningen, The Netherlands

Abstract

We investigate intersection types and resource lambda-calculus in deep-inference proof theory. We give a unified type system that is parametric in various aspects: it encompasses resource calculi, intersection-typed lambda-calculus, and simply-typed lambda-calculus; it accommodates both idempotence and non-idempotence; it characterizes strong and weak normalization; and it does so while allowing a range of algebraic laws to determine reduction behaviour, for various quantitative effects. We give a parametric resource calculus with explicit sharing, the "collection calculus", as a Curry–Howard interpretation of the type system, that embodies these computational properties.

2012 ACM Subject Classification Theory of computation → Proof theory; Theory of computation → Lambda calculus

Keywords and phrases Lambda-calculus, Deep inference, Intersection types, Resource calculus

Digital Object Identifier 10.4230/LIPIcs.CSL.2021.33

Funding This work was supported by EPSRC Project EP/R029121/1 *Typed lambda-calculi with sharing and unsharing*

Acknowledgements We would like to thank Ugo Dal Lago, Delia Kesner, Luc Pelissier, Nicolas Wu, and the anonymous referees for their constructive engagement with our work.

1 Introduction

Of the various qualitative and quantitative approaches to λ-calculus, which include intersection types [15, 16, 23], resource calculi [9, 30, 21], and relational models [31, 33], many are known to be related, often in deep and interesting ways. We are curious if there is a common foundation, a question that we approach through deep-inference proof theory. Here, we give a unified, structural perspective on *intersection types* and *resource calculi*, in the form of a deep quantitative proof system. It is both a simple type system for a resource calculus, the *collection calculus* that we introduce here, and an intersection type system for an explicit-substitution calculus, the *structural λ-calculus* [3, 4]. In both cases, it can be parameterized in various algebraic laws to obtain different quantitative effects.

The computational side of deep inference

Deep inference, as a family of proof formalisms, has remarkable properties: quasi-polynomial proof complexity and normalization for propositional classical logic [28, 12], non-elementary proof compression for first-order classical logic [5], and the ability to express logics for which no sequent calculus can exist [39], among others. It is a natural question if such striking features can be put to computational use. In previous work in this direction, the second author and co-authors derived two *atomic* λ-calculi, which characterize different versions of *full laziness*, from the duplication properties of intuitionistic deep inference [26, 37].

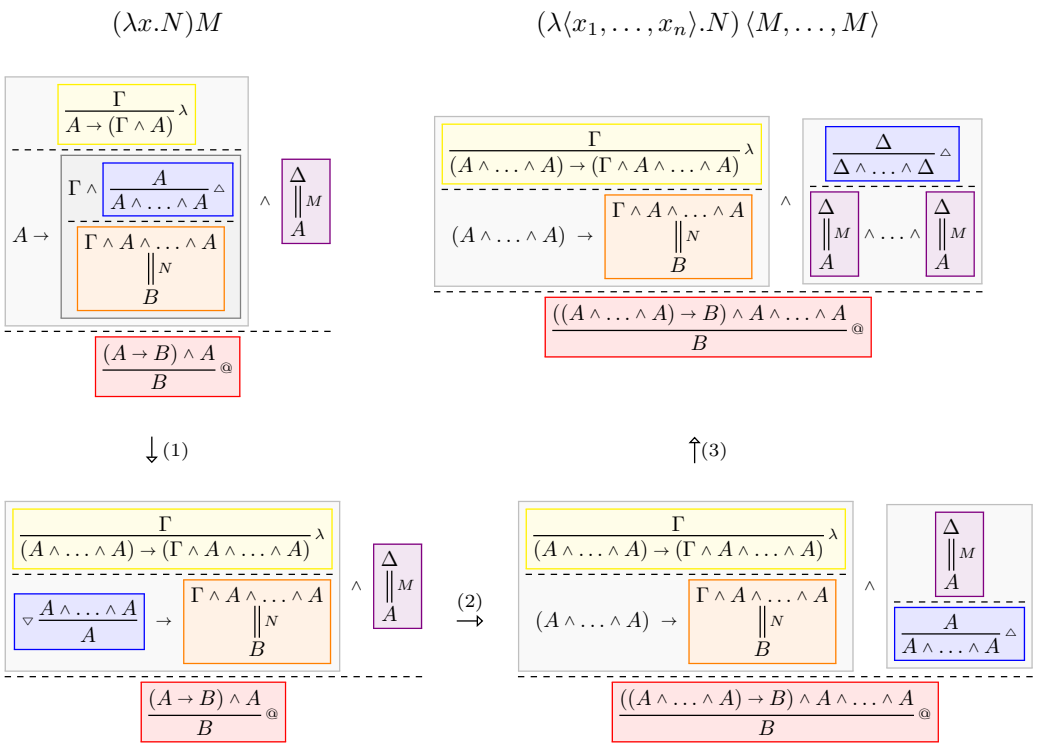


Figure 1 *Deriving resource calculus by proof transformations.* The derivation top left is for the lambda-term $(\lambda x.N)M$. The blue contraction rule ($\triangle$) passes through the yellow abstraction rule (λ) in step (1), ending up as the inverted rule ($\triangledown$) to reflect that ($\to$) reverses "polarity" on the left; it then passes through the application rule ($@$) in step (2); and duplicates the argument derivation M in step (3). The resulting derivation, top right, is for an interpretation of the original term as a resource term $(\lambda\langle x_1,\ldots,x_n\rangle.N)\,\langle M,\ldots,M\rangle$, where the variables x_i represent the occurrences of x. Categorically, (λ) and ($@$) are the η and ϵ transformations of the adjunction between ($\to$) and ($\wedge$), while ($\triangle$) is the diagonal map for ($\wedge$); the steps (1)–(3) then reflect the (di)naturality of these maps, and the inversion of ($\triangle$) to ($\triangledown$) reflects the contravariance of ($\to$) in its first argument.

In this paper, we investigate *intersection types* [15] and *resource lambda-calculi* [9] from the perspective of deep inference. We will work in the formalism *open deduction* [25]; see Section 3 for an introduction. We start by observing that in lambda-calculus and natural deduction, duplication and beta-reduction are intimately related: in an abstraction $\lambda x.N$, the bound occurrences of x in N represent a potential duplication, which can only be effected by a beta-step on $(\lambda x.N)M$. Systems like sequent calculi, proof nets, and explicit-substitution calculi may separate beta-reduction and duplication by an explicit *contraction* rule. Deep inference goes one step further: contraction rules may *pass through* other proof rules. We illustrate this in Figure 1. For a simply-typed open-deduction proof, one may carry out all latent duplication by pushing the contractions through the proof in the way of the example, until they disappear at the top or bottom of the proof. Doing so transforms a simple type derivation for a lambda-term into a (non-idempotent) intersection-type derivation, or equivalently into a simple type derivation for a corresponding resource term. The result is familiar from resource calculi, which may unfold the redex $(\lambda x.N)M$ to $(\lambda\langle x_1,\ldots,x_n\rangle.N)\langle M,\ldots,M\rangle$ where the variables x_i represent the n occurrences of x in N. Crucially, in open deduction this transformation applies not only to redexes, but to individual abstractions and applications.

In Figure 1, the conjunction ($\wedge$) has two distinct rôles: its standard rôle in the application rule, in typing NM, plus that of creating a *collection* of derivations, in typing $\langle M, \dots, M \rangle$. We separate both rôles by introducing an intersection type operator ($+$) for collections, leaving the conjunction in its traditional rôle. We give the operator ($+$), and the rules for it to interact with ($\wedge$), below: (1) as is characteristic of connectives in open deduction, ($+$) applies to *derivations* as well as formulas, giving a derivation from $A+C$ to $B+D$; (2) the *contraction* rule ($\triangle$) is modified to transform a conjunction into an intersection; and (3) conjunction and intersection are interchanged by a (non-invertible) *medial* rule (m).

$$(1)\quad \begin{matrix} A \\ \| \\ B \end{matrix} + \begin{matrix} C \\ \| \\ D \end{matrix} \qquad (2)\quad \frac{A+B}{A \wedge B}\,\triangle \qquad (3)\quad \frac{(A+B)\wedge(C+D)}{(A\wedge C)+(B\wedge D)}\,\mathsf{m}$$

Our construction makes essential use of the characteristic properties of open deduction. Firstly, operators apply to *derivations* as well as *formulas*, as in (1) above. Via the Curry–Howard correspondence this gives us a natural, simultaneous treatment of collections of *terms* and collections of *types*, giving a tight correspondence between resource terms and their type derivations. Secondly, *medial*-style rules [11, 38, 6] are unique to deep-inference systems, and are at the root of many of the contributions of the theory, including the complexity results for classical logic mentioned above, and both atomic lambda-calculi.

The most salient feature of our approach is that the calculus and the type system can be *parameterized* in various algebraic laws, which captures for instance the familiar distinction between *idempotent* and *non-idempotent* intersection types. This is made possible by our structural approach: once the above constructions (1–3) are available, the proof system is in principle agnostic about the further properties of collections.

Our technical exposition starts with a system of simple types in open deduction, in Section 3, for the *Structural λ-Calculus* $\lambda\mathsf{j}$ of Accattoli and Kesner [3, 4], which we recall in Section 2 and here abbreviate SC. The work of Accattoli and Kesner derives the SC, and the related *Linear Substitution Calculus* (LSC, [1]), from an extensive search for good reduction properties in explicit-substitution calculi, inspired by linear logic. Here, we observe that the SC also arises as a natural Curry–Howard-style interpretation of intuitionistic open deduction. We view this as further support for our proof-theory based approach. A version of the SC with linear use of variables was the basis for both atomic λ-calculi [26, 37].

A precursor to the present work is the workshop paper [27]. Proofs are in the Appendix.

Related work Intersection types, in their idempotent variant, have been studied to characterize several kinds of normalization [15, 16, 36]. The non-idempotent variant introduced in [23] is strictly related to linear logic [18, 19] and induces a well-known denotational model of the λ-calculus and linear logic: relational semantics [13, 34]. The literature about intersection types is huge, let us mention [14] for a survey and [2] for recent developments.

Resource-sensitive calculi [9, 30] can be seen as a "dynamic" counterpart of non-idempotent intersection types, often inspired by linear logic, see for instance [21, 22, 33].

Approaches to resource calculi and intersection types from a proof-theoretic perspective are uncommon; for the former, since qualitative and quantitative properties are already captured through the *term calculus*, and for the latter since the restriction that intersection types can only be formed for proofs *of the same term* is a fundamental departure from traditional logical systems; exceptions to the latter are [35, 20].

A different unified perspective, via category theory, is given in [32]; the conceptual difference is that their approach is *extensional* (characterizing qualitative systems through their properties) where ours is *intensional* (we give an underlying syntactic structure).

2 The structural λ-calculus

Our point of departure is the *structural λ-calculus* (SC) of Accattoli and Kesner [3, 4], an explicit-substitution λ-calculus where closures are evaluated by *decomposition* and *linear substitution*. This puts the dynamic behaviour of the calculus away from implicit substitution, and closer to *graph reduction*; we prefer to call it an *explicit-sharing* calculus instead.

As an interpretation of deep inference, other calculi with explicit sharing would be equally suitable, such as λlxr of Kesner and Lengrand [29]; we choose the SC for its concise notation.

▶ **Definition 1.** *The* terms r, s, t *of the SC are defined by the grammar*

$$r, s, t \ ::= \ x \ \mid \ ts \ \mid \ \lambda x.t \ \mid \ t[x \leftarrow s]$$

with from left to right: a variable*; an* application*; an* abstraction*, which binds x in t; and a* closure*, which binds x in t.*

We call $[x \leftarrow s]$ a *sharing*, abbreviated to $[\phi]$, and write $[\Phi]$ for a sequence of sharings $[x_1 \leftarrow s_1] \dots [x_n \leftarrow s_n]$, or $t[\Phi]$ when applied as closures to a term t. We write $\{t/x\}$ for the (capture-avoiding) substitution of t for x, and $|t|_x$ for the number of free occurrences of x in t. The set of free variables of a term t is denoted by $\mathsf{fv}(t)$.

▶ **Definition 2.** *The* reduction rules *of the SC are the contextual closure of the rules below.*

$$\begin{array}{rcll} (\lambda x.t)[\Phi]s & \to_{\mathsf{b}} & t[x \leftarrow s][\Phi] & \text{(beta)} \\ t\{x/y\}[x \leftarrow s] & \to_{\mathsf{c}} & t[x \leftarrow s][y \leftarrow s] & |t|_x, |t|_y \geq 1 \quad \text{(copy)} \\ t[x \leftarrow s] & \to_{\mathsf{e}} & t\{s/x\} & |t|_x = 1 \quad \text{(evaluate)} \\ t[x \leftarrow s] & \to_{\mathsf{d}} & t & |t|_x = 0 \quad \text{(delete)} \end{array}$$

We set $\to_{\neg\mathsf{b}} = \to_{\mathsf{c}} \cup \to_{\mathsf{d}} \cup \to_{\mathsf{e}}$ *and* $\to_{\mathsf{sc}} = \to_{\mathsf{b}} \cup \to_{\neg\mathsf{b}}$.

The *beta*-rule includes the closures $[\Phi]$ so that these do not block the redex: it acts *at a distance*. This mimicks graph reduction and obviates the need to permute closures. In the *copy* rule, the notation $t\{x/y\}$ is used to separate the occurrences of x into two (non-empty) classes: those that occur as x in t and those that occur as y in t. The sharing $[x \leftarrow s]$ can then be duplicated and split among them. This is used to isolate a variable with one occurrence, to which the *evaluate* rule then applies, whose substitution $\{s/x\}$ is *linear*.

For a rewrite relation $\to$, we write $\twoheadrightarrow$ for its reflexive-transitive closure, and $\to\!\!\!\!\!\twoheadrightarrow$ for reduction to normal form. The λ-calculus embeds into the SC without using a translation, as λ-terms are the SC-terms that do not have closures. The *unfolding* $t^\bullet$ of a term t, defined below, evaluates all closures by substitutions, which interprets SC-terms as λ-terms.

$$x^\bullet = x \qquad \lambda x.t^\bullet = \lambda x.t^\bullet \qquad (ts)^\bullet = t^\bullet s^\bullet \qquad (t[x \leftarrow s])^\bullet = t^\bullet\{s^\bullet/x\}.$$

▶ **Proposition 3** (Simulations)**.** *Let t be a SC-term and s be a λ-term.*

1. From SC to λ-calculus*: If $t \to_{\mathsf{b}} t'$ then $t^\bullet \to_\beta t'^\bullet$; if $t \to_{\neg\mathsf{b}} t'$ then $t^\bullet = t'^\bullet$.*
2. From λ-calculus to SC*: If $s \to_\beta s'$ then $s \to_{\mathsf{b}} \twoheadrightarrow_{\neg\mathsf{b}} s'$.*

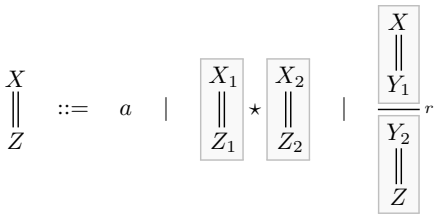
▶ **Proposition 4** (Collated results from [4])**.** *The SC has the following key properties.*

1. *The normal forms of $\to_{\neg\mathsf{b}}$ are exactly the λ-terms.*
2. *The normal forms of $\to_{\mathsf{sc}}$ are exactly the β-normal λ-terms.*
3. *For any SC-term t, one has $t \to\!\!\!\!\!\twoheadrightarrow_{\neg\mathsf{b}} t^\bullet$; in particular, $t = t^\bullet$ for any λ-term.*
4. *The relations $\to_{\mathsf{b}}$, $\to_{\neg\mathsf{b}}$, and $\to_{\mathsf{sc}}$ are confluent; $\to_{\mathsf{b}}$ and $\to_{\neg\mathsf{b}}$ are strongly normalizing.*
5. *Preservation of strong normalization: if a λ-term t has an infinite $\to_{\mathsf{sc}}$-reduction, then it has an infinite $\to_\beta$-reduction.*

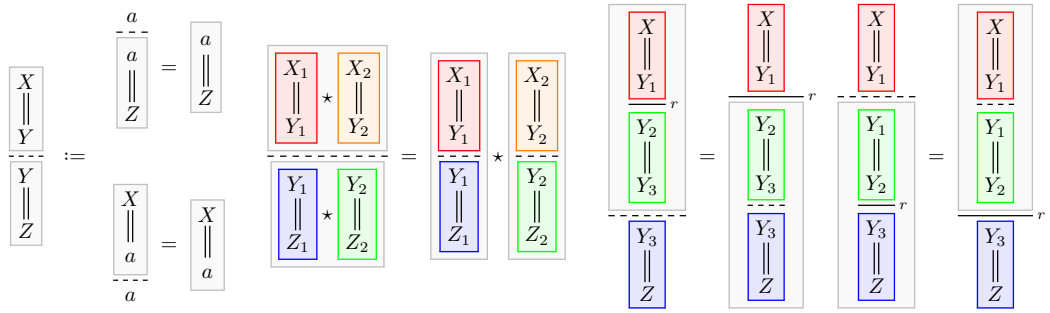
3 A deep type system

Open deduction is a dialect of deep-inference proof theory, introduced by Guglielmi, Gundersen, and Parigot [25], where proofs are constructed in two directions: *horizontally* by *connectives*, and *vertically* by *rules*. We give a brief formal introduction.

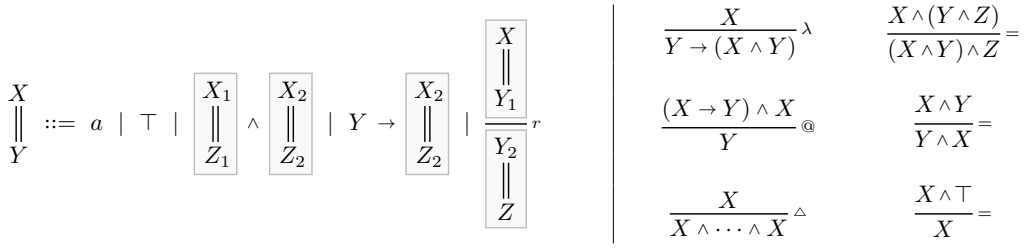
A *derivation* from a *premise* formula X to a *conclusion* formula Z is constructed inductively as below, with from left to right: a propositional atom a, where $X = Z = a$; *horizontal construction* with a connective $\star$, where $X = X_1 \star X_2$ and $Z = Z_1 \star Z_2$; and *vertical construction* with an inference rule r from Y_1 to Y_2. Boxes serve as parentheses (since derivations extend in two dimensions) and may be omitted.



Derivations are considered up to associativity of vertical construction. One may consider *formulas* as derivations that omit vertical construction. The binary $\star$ may be generalized to 0-ary, unary, and n-ary operators, and it may have *negative* arguments where a derivation becomes *inverted*, exchanging premise and conclusion, such as to the left of an implication—though we will avoid the need for these. *Composition* of a derivation from X to Y and one from Y to Z, depicted by a dashed line, is a defined operation:



We specialize the above to a proof system for conjunction-implication intuitionistic logic, similar to that of Brünnler and McKinley [10], through the grammar and inference rules below. Note the inclusion of the unit $\top$ as a 0-ary operator, and the restriction of the left subderivation of $(\to)$ to a formula, to avoid introducing inverted derivations. The rules are: *abstraction* (λ), *application* $(@)$, and n-ary *contraction* $(\triangle)$ on the left and the invertible rules for *associativity*, *symmetry*, and *unitality* of conjunction on the right. A 0-ary contraction, with conclusion $\top$, is a *weakening*. We will leave the invertible rules implicit in derivations, and consider conjunction modulo associativity and unitality.



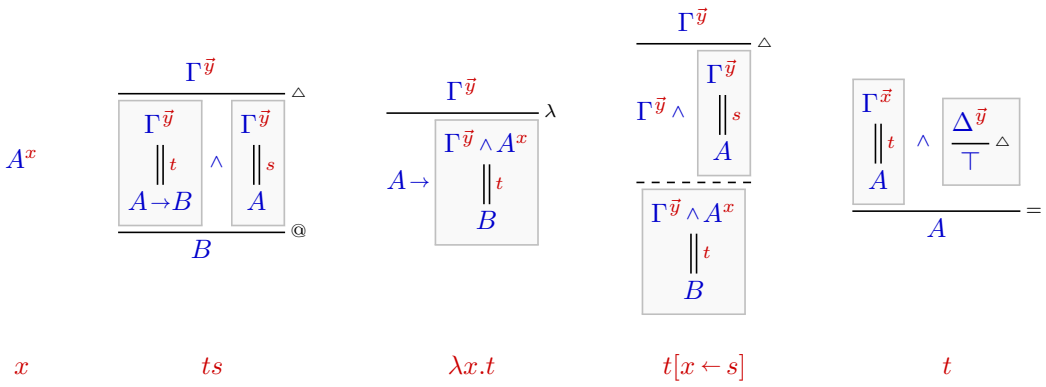
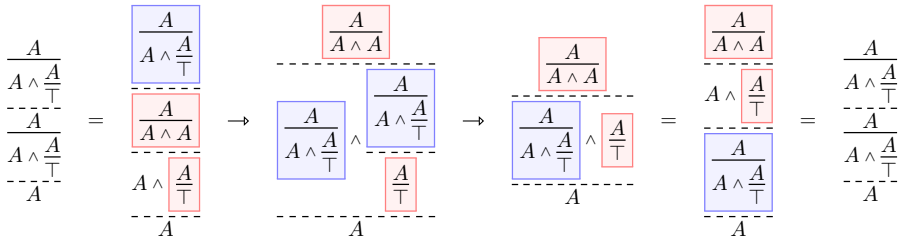


Figure 2 *An open-deduction system of simple types for the structural λ-calculus.* In ts, both t and s are given the same context $\Gamma^{\vec{y}}$ with $\vec{y} = \mathsf{fv}(t) \cup \mathsf{fv}(s)$ by applying the rules $\triangle$ and $=$ as in the rightmost construction, where $\vec{x} \cap \vec{y} = \varnothing$; and similarly for $t[x \leftarrow s]$.

Typing the structural λ-calculus

We give an open-deduction system of simple types for the SC. Not all derivations correspond to a term: the calculus picks out a subset of derivations, imposes certain equivalences, and guides reduction. The latter is essential, since naïve reduction in the proof system creates cycles of contractions duplicating each other: this example is from [10]—see there for detail.

▶ **Example 5** ([10])**.** The natural reductions for a contraction or weakening rule are to duplicate respectively to delete the derivation above it (see also Figure 3). Applied naïvely, this reduction is non-terminating by the example below (all explicit rules are contractions).



The type system is in Figure 2. A term t is typed by a derivation from Γ to A, which we indicate as below left. The *structured types* A and Γ are respectively a *basic type* and a *context type*, generated by the respective grammars below. A context type $\Gamma = A_1 \wedge \cdots \wedge A_n$ in the premise of the derivation for t types a vector of *context variables* $\vec{x} = x_1, \ldots, x_n$, which include the free variables of t, but may be expanded to include variables not occurring in t via the rightmost derivation in Figure 2. We make context variables explicit as $\Gamma^{\vec{x}} = A_1^{x_1} \wedge \cdots \wedge A_n^{x_n}$.

A derivation for t: Γ $\|t$ A

Basic types: $A, B, C, D \;::=\; a \mid A{\to}B$

Context types: $\Gamma, \Delta, \Lambda, \Sigma \;::=\; \top \mid A \mid \Gamma \wedge \Delta$

In the calculus, contraction is implicit via the use of variables (and made explicit in the reduction rules by considering variable *occurrences*). Correspondingly, we consider derivations modulo the equivalences ($\triangle\triangle$) and ($\frac{\triangle}{\triangle}$) below right, where in ($\triangle\triangle$) both contractions have the same width (the same number of formulas X respectively Y in the conclusion).

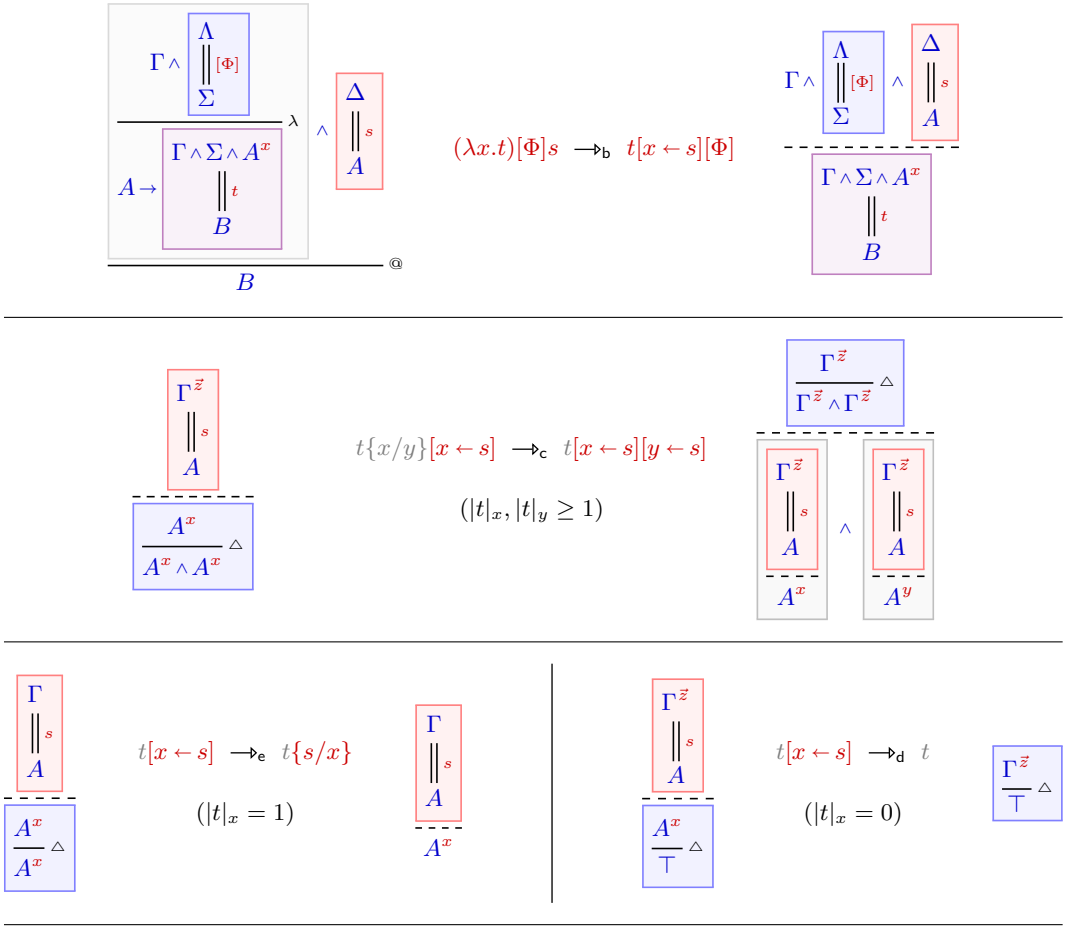
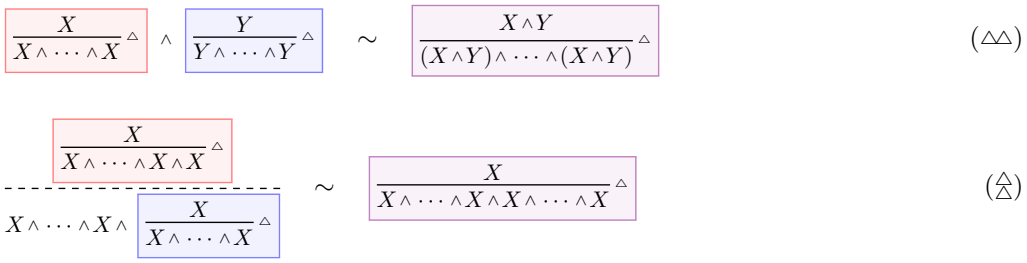
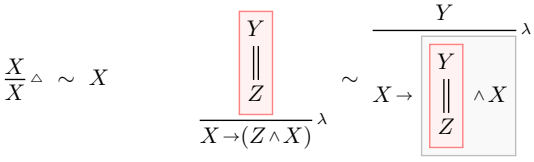


Figure 3 Subject reduction for the simply-typed structural λ-calculus



We consider the unary contraction equivalent to an identity (below left), though we may choose to deploy it as a "marker" to differentiate between an *explicit* substitution $t[x \leftarrow s]$ where $|t|_x = 1$ (*with* unary contraction) and the *implicit* substitution $t\{s/x\}$ to which it reduces (*without*). We include a naturality equation for the abstraction (below right), to capture the equation $(\lambda x.t)\{s/z\} = \lambda x.t\{s/z\}$ (where $x \neq z$) for subsitution.



Typing derivations for the reduction rules are given in Figure 3. For the rules $(\mathsf{c}, \mathsf{d}, \mathsf{e})$ we omit the term t from the derivations, for brevity. The figure witnesses that:

▶ **Proposition 6** (Subject reduction). *SC reduction preserves typing.*

4 The collection calculus

We extend the SC with an abstract notion of *collection*, applied to terms, types, and derivations. The resulting *collection calculus* (CC) is a resource λ-calculus that is parameterised in a specific choice of collection, such as sets, multisets, or with a minor modification, sequences. We generate collections syntactically, by combining empty $\langle\rangle$ and singleton $\langle t\rangle$ collections with a binary *append* operator $+$, and then consider these modulo standard algebraic laws.

▶ **Definition 7.** *The* collection calculus *(CC) is given by the* terms *and* collection terms*:*

$$r, s, t \;::=\; x \;\mid\; t\tau \;\mid\; \lambda x.t \;\mid\; t[x \leftarrow \tau] \qquad\qquad \rho, \sigma, \tau \;::=\; \langle\rangle \;\mid\; \langle t\rangle \;\mid\; \sigma + \tau$$

where terms are as for the SC, and collection terms are empty*,* singleton*, and* append*.*

The collection calculus is *parameterized* in a *collection algebra*, a preorder $(\leq)$ over collection terms generated by a selection of algebraic equalities and inequalities, which governs the behaviour of collections under reduction. A reduction step on a closure $t[x \leftarrow \tau]$ will treat the collection τ modulo $(\leq)$: it will correspond to a (non-deterministic) syntactic reduction on $t[x \leftarrow \sigma]$ for a chosen σ such that $\tau \leq \sigma$. Conceptually, $(\leq)$ implements the *structural* aspects of reduction, such as duplication, deletion, and exchange.

The algebraic laws are the following, where $\sigma = \tau$ means that both $\sigma \leq \tau$ and $\tau \leq \sigma$. Commensurate with the intuition of $(\leq)$ as a reduction relation, it satisfies *reflexivity* $(\tau \leq \tau)$, *transitivity* (if $\rho \leq \sigma$ and $\sigma \leq \tau$ then $\rho \leq \tau$), and *contextual closure* (if $\rho \leq \sigma$ then $\tau + \rho \leq \tau + \sigma$ and $\rho + \tau \leq \sigma + \tau$). The current presentation further assumes *associativity*, *unitality*, and *symmetry* (below), so that collections are *multisets*. Relaxing these laws will be discussed in Section 7. The laws of *redundancy*, *duplicability*, and *idempotence* are optional parameters.

Associativity	$\rho+(\sigma+\tau) \;=\; (\rho+\sigma)+\tau$	**Redundancy**	$\tau \;\leq\; \langle\rangle$
Unitality	$\langle\rangle+\tau \;=\; \tau \;=\; \tau+\langle\rangle$	**Duplicability**	$\tau \;\leq\; \tau+\tau$
Symmetry	$\sigma+\tau \;=\; \tau+\sigma$	**Idempotence**	$\tau \;=\; \tau+\tau$

Reduction is non-deterministic, and produces an (idempotent) formal sum of terms at the meta-level, distinct from collection terms and the append operator.

▶ **Definition 8.** *The* reduction rules *of the CC are the contextual closure of the rules below.*

$$(\lambda x.t)[\Phi]\tau \;\to_{\mathsf{b}}\; t[x \leftarrow \tau][\Phi] \tag{beta}$$

$$t\{x/y\}[x \leftarrow \tau] \;\to_{\mathsf{c}}\; \sum_{\tau \leq \rho+\sigma} t[x \leftarrow \rho][y \leftarrow \sigma] \qquad |t|_x, |t|_y \geq 1 \tag{copy}$$

$$t[x \leftarrow \tau] \;\to_{\mathsf{e}}\; \sum_{\tau \leq \langle s\rangle} t\{s/x\} \qquad |t|_x = 1 \tag{evaluate}$$

$$t[x \leftarrow \tau] \;\to_{\mathsf{d}}\; \sum_{\tau \leq \langle\rangle} t \qquad |t|_x = 0 \tag{delete}$$

We set $\to_{\neg\mathsf{b}} = \to_{\mathsf{c}} \cup \to_{\mathsf{d}} \cup \to_{\mathsf{e}}$*, and* $\to_{\mathsf{cc}} = \to_{\mathsf{b}} \cup \to_{\neg\mathsf{b}}$*.*

Observe that for a closure $t[x \leftarrow \tau]$, the number of occurrences $|t|_x$ determines which reduction step applies, while the collection algebra ($\leq$) determines what reducts are obtained. For the *evaluate* and *delete* steps, the sum implements the possibility of *deadlock*: the result is either a singleton or the empty sum 0. The laws of *redundancy* and *duplicability* allow deletion respectively duplication of the terms in a collection.

▶ **Example 9.** We have the following $\neg\mathsf{b}$-reductions, writing $\langle t_1, \dots, t_n \rangle$ for $\langle t_1 \rangle + \dots + \langle t_n \rangle$, where the row (1) gives the reducts for the plain collection algebra (collections as multisets); (2) gives those with *redundancy*; (3) those with *duplicability*; and (4) those with both laws.

$x\langle x \rangle[x \leftarrow \langle s, t \rangle] \twoheadrightarrow_{\neg\mathsf{b}}$	$x[x \leftarrow \langle s, t \rangle] \twoheadrightarrow_{\neg\mathsf{b}}$	$x\langle x \rangle[x \leftarrow \langle s \rangle] \twoheadrightarrow_{\neg\mathsf{b}}$	
$s\langle t \rangle + t\langle s \rangle$	0	0	(1)
$s\langle t \rangle + t\langle s \rangle$	$s + t$	0	(2)
$s\langle t \rangle + t\langle s \rangle$	0	$s\langle s \rangle$	(3)
$s\langle s \rangle + s\langle t \rangle + t\langle s \rangle + t\langle t \rangle$	$s + t$	$s\langle s \rangle$	(4)

Traditionally, intersection type systems have featured *idempotence* instead of *duplicability*, rendering collections as *sets*. While natural from an algebraic perspective, *duplicability* and *redundancy* are a closer match with the reduction behaviour of contraction and weakening rules. The missing direction is also derived through *redundancy* and *unitality*: $\tau + \tau \leq \tau + \langle\rangle = \tau$.

With *duplicability* or *idempotence*, the *copy* reduction step produces infinite sums, since the class of collections $\{\sigma + \rho \mid \tau \leq \sigma + \rho\}$ is infinite as soon as τ is non-empty. However, most duplication may safely be delayed. Writing $\sigma \subseteq \tau$ for the sub-multiset relation (i.e. each element of σ occurs at least as many times in τ), the *copy* rule may be restricted to those reducts where $\rho, \sigma \subseteq \tau$. Likewise, with *redundancy*, deletion may be delayed and relegated to *evaluate* and *delete* steps, restricting *copy* to reducts where $\tau \subseteq \rho + \sigma$. And with both laws, we only need consider $\rho, \sigma = \tau$.

▶ **Example 10.** We consider the reduction from $(\lambda z.z\langle z \rangle)\langle x, x, y \rangle$, which starts as follows.

$$(\lambda z.z\langle z \rangle)\langle x, x, y \rangle \quad \to_{\mathsf{b}} \quad z\langle z \rangle[z \leftarrow \langle x, x, y \rangle] \quad \to_{\mathsf{c}} \quad \sum_{\langle x,x,y \rangle \leq \rho + \sigma} w\langle z \rangle[w \leftarrow \rho][z \leftarrow \sigma]$$

With the plain collection algebra, this sum consists of:

$$\begin{array}{ccccccc} & w\langle z \rangle[w \leftarrow \langle\rangle][z \leftarrow \langle x, x, y \rangle] & + & w\langle z \rangle[w \leftarrow \langle x \rangle][z \leftarrow \langle x, y \rangle] & + & w\langle z \rangle[w \leftarrow \langle y \rangle][z \leftarrow \langle x, x \rangle] \\ + & w\langle z \rangle[w \leftarrow \langle x, x, y \rangle][z \leftarrow \langle\rangle] & + & w\langle z \rangle[w \leftarrow \langle x, y \rangle][z \leftarrow \langle x \rangle] & + & w\langle z \rangle[w \leftarrow \langle x, x \rangle][z \leftarrow \langle y \rangle] \end{array}$$

This reduces to zero, since no summand has both ρ and σ as singletons. With *redundancy*, reduction continues as below left—by delaying deletion as discussed previously, the above *copy* step is not affected, and recall that the meta-level sum is idempotent. With only *idempotence*, both occurrences of x may be collapsed and reduction instead continues as below right. While we need to consider additional reducts for the *copy* rule above, such as $w\langle z \rangle[w \leftarrow \langle x, y \rangle][z \leftarrow \langle y \rangle]$, these do not add normal forms, since x and y may not be deleted. With also *redundancy*, $x\langle x \rangle$ or $y\langle y \rangle$ are added as normal forms.

redundancy: $\dots \twoheadrightarrow_{\mathsf{e}} x\langle x \rangle + x\langle y \rangle + y\langle x \rangle$ idempotence: $\dots \twoheadrightarrow_{\mathsf{e}} x\langle y \rangle + y\langle x \rangle$

Reduction does not produce $x\langle x, y \rangle$ or $y\langle x, y \rangle$, nor does it produce $x\langle\rangle$ or $y\langle\rangle$. By contrast, given *idempotence*, $z\langle z, z \rangle[z \leftarrow \langle x, x, y \rangle]$ *does* (non-deterministically) reduce to $x\langle x, y \rangle$ and $y\langle x, y \rangle$. The meaning of the algebraic laws is not to equate terms, as idempotence would

$z\langle z\rangle$ and $z\langle z,z\rangle$, but to govern the behaviour of collections under reduction. This is standard for resource calculi, and the alternative would severely complicate reduction: evaluating a closure $t[x \leftarrow \tau]$ would require duplication within any collection σ in t where $x \in \mathsf{fv}(\sigma)$.

▶ **Proposition 11.** *The normal forms of* $\twoheadrightarrow_{\neg\mathsf{b}}$ *and* $\twoheadrightarrow_{\mathsf{cc}}$ *are (non-deterministic sums over) terms of the form* s_0 *respectively* t_0 *given as follows.*

$$s_0 ::= x \mid s_0\langle s_0,\dots,s_0\rangle \mid \lambda x.s_0 \qquad t_1 ::= x \mid t_1\langle t_0,\dots,t_0\rangle \qquad t_0 ::= t_1 \mid \lambda x.t_0$$

▶ **Definition 12.** *The* unfolding $t^\bullet$ *of a CC-term* t *and* substitution *for collections* $\{\tau/x\}$ *are defined as follows.*

$$\begin{aligned} x^\bullet &= x \\ (t\tau)^\bullet &= t^\bullet\tau^\bullet \\ (\lambda x.t)^\bullet &= \lambda x.t^\bullet \\ (t[x\leftarrow\tau])^\bullet &= t^\bullet\{\tau^\bullet/x\} \\ \langle t_1,\dots,t_n\rangle^\bullet &= \langle t_1^\bullet,\dots,t_n^\bullet\rangle \end{aligned}$$

$$\begin{aligned} x\{\sigma/x\} &= \textstyle\sum_{\sigma\le\langle s\rangle} s \\ x\{\sigma/y\} &= \textstyle\sum_{\sigma\le\langle\rangle} x \qquad (\text{if } x\neq y) \\ (t\tau)\{\sigma/y\} &= \textstyle\sum_{\sigma\le\sigma_1+\sigma_2} (t\{\sigma_1/y\})(\tau\{\sigma_2/y\}) \\ (\lambda x.t)\{\sigma/y\} &= \lambda x.(t\{\sigma/y\}) \\ t[x\leftarrow\tau]\{\sigma/y\} &= \textstyle\sum_{\sigma\le\sigma_1+\sigma_2} t\{\sigma_1/y\}[x\leftarrow\tau\{\sigma_2/y\}] \\ \langle\rangle\{\sigma/y\} &= \textstyle\sum_{\sigma\le\langle\rangle}\langle\rangle \\ \langle t\rangle\{\sigma/y\} &= \langle t\{\sigma/y\}\rangle \\ (\tau_1+\tau_2)\{\sigma/y\} &= \textstyle\sum_{\sigma\le\sigma_1+\sigma_2} \tau_1\{\sigma_1/y\}+\tau_2\{\sigma_2/y\} \end{aligned}$$

A single closure $t[x \leftarrow \tau]$ is evaluated by a substitution $t\{\tau/x\}$, and the unfolding of a term evaluates all closures, commensurate with $\neg\mathsf{b}$-reduction to normal form in a way that we make precise below.

▶ **Proposition 13.** $\neg\mathsf{b}$*-Reduction of a CC-term* t *is strongly normalizing, and confluent in the following sense: if* $t \twoheadrightarrow_{\neg\mathsf{b}} \sum_{s\in S} s$ *and* $t^\bullet = \sum_{r\in R} r$ *then* $S \subseteq R$.

▶ **Proposition 14.** *Without idempotence and duplicability, CC-terms are strongly normalizing.*

A main purpose of resource calculi is to provide quantitative bounds on the length of reduction sequences. We will measure the length of *non-deterministic* reduction paths, which select only one term from a formal sum of reducts, by a *reduction weight* $|t|$ derived from the constructors in a term t. With the plain collection algebra, the number of *beta* steps is bounded by the number of abstractions and applications. For non-beta steps, a reduction path $t[x \leftarrow \tau] \twoheadrightarrow_{\neg\mathsf{b}} t\{\tau/x\}$ where $|t|_x = n$ consists of $2n-1$ steps ($n-1$ *copy* steps and n *evaluate* steps) if $n \geq 1$, or one *delete* step if $n = 0$. This gives the constraints that variables contribute 2 to the reduction weight, weakenings contribute 1, and sharings otherwise contribute -1. Then $|t|$ is defined as follows, where $x \notin \mathsf{fv}(r)$ but $x \in \mathsf{fv}(s)$:

$$|x| = 2 \qquad |t\tau| = |t| + |\tau| \qquad |\lambda x.r| = |r| + 2 \qquad |\lambda x.s| = |s| \qquad |r[x\leftarrow\tau]| = |r| + |\tau| + 1 \qquad |s[x\leftarrow\tau]| = |s| + |\tau| - 1 \qquad |\langle t_1,\dots,t_n\rangle| = \sum_{i=1}^{n} |t_i|$$

In the absence of further algebraic laws, quantitative bounds are *exact*; with *redundancy*, they are *upper bounds*; and with *duplicability*, they are *lower bounds*.

▶ **Proposition 15.** *The length of a (non-deterministic) reduction sequence* $s \twoheadrightarrow_{\mathsf{cc}} t$ *is:*

1. *without algebraic laws, exactly* $|s| - |t|$*;*
2. *with only* redundancy*, at most* $|s| - |t|$*;*
3. *with only* duplicability*, at least* $|s| - |t|$.

5 A deep quantitative type system

Figure 4 gives an open-deduction proof system for a logic of quantitative types. Unlike the two-sorted CC, which has separate sorts for terms and collection terms, the type system is single-sorted, and the constructors *empty* $\langle\rangle$ and *append* $+$ are included with the regular types and derivations. The inference rules include a modified n-ary *contraction* ($\triangle$), with the 0-ary case given below, the $m{\times}n$-ary *medial* (m), specialized to the dimensions $0{\times}0$, $2{\times}0$, $1{\times}1$, $0{\times}2$, and $2{\times}2$ below, and the rule ($\leq$) that implements the algebraic laws for types. The inequality ($\leq$) as a typing rule represents a generalized *structural* rule, generalizing the *weakening*, *contraction*, and *exchange* rules familiar from sequent calculi and other proof systems. A similar algebraic rule appears in the intersection type system of [7].

$$\frac{\langle\rangle}{\top}\,\triangle \qquad \frac{\top}{\langle\rangle}\,\mathsf{m} \qquad \frac{\langle\rangle\wedge\langle\rangle}{\langle\rangle}\,\mathsf{m} \qquad \frac{X}{X}\,\mathsf{m} \qquad \frac{\top}{\top+\top}\,\mathsf{m} \qquad \frac{(W+X)\wedge(Y+Z)}{(W\wedge Y)+(X\wedge Z)}\,\mathsf{m}$$

The operators *append* and *empty* are parameterized by the same algebraic laws as for collection terms: we assume *associativity*, *unitality*, and *symmetry*, though these can be relaxed without fear of inconsistency, and optional are *redundancy*, *duplicability*, and *idempotence*.

Associativity	$A+(B+C) = (A+B)+C$	**Redundancy**	$A \leq \langle\rangle$
Unitality	$\langle\rangle+A = A = A+\langle\rangle$	**Duplicability**	$A \leq A+A$
Symmetry	$A+B = B+A$	**Idempotence**	$A = A+A$

Figure 5 presents a type system for the collection calculus within the open-deductive quantitative system. A term t is typed by a derivation over *structured* types, defined below, with as conclusion a *basic type* A and as premise a *context type* Γ, which is itself a conjunction over *collection types* I. The premise Γ of a derivation for t types a vector of *context variables* $\vec{x}$ that includes the free variables of t, made explicit by writing $\Gamma^{\vec{x}} = I_1^{x_1}\wedge\cdots\wedge I_n^{x_n}$.

A derivation for t: $\dfrac{\Gamma^{\vec{x}}}{A}\,t$

Basic types:	A,B,C,D	$::= a \mid I{\to}A$
Collection types:	I,J,K,L	$::= \langle\rangle \mid A \mid I+J$
Context types:	$\Gamma,\Delta,\Lambda,\Sigma$	$::= \top \mid I \mid \Gamma\wedge\Delta$

In typing a collection of terms $\tau = \langle t_1,\dots,t_n\rangle$, the medial generates the type of each context variable x, by combining the types I_1 through I_n for x in each t_i into the type $I_1+\cdots+I_n$. For convenience, we capture the effect of the $n{\times}0$-medial by abbreviating contexts of empty collections by $\Gamma^{\vec{x}}_{\langle\rangle} = \langle\rangle^{x_1}\wedge\cdots\wedge\langle\rangle^{x_n}$, and that of the $n{\times}2$-medial by an operator $(+\!\!+)$:

$$\frac{\Gamma^{\vec{x}}_{\langle\rangle}}{\langle\rangle}\,\mathsf{m} \qquad \frac{(\Gamma+\!\!+\Delta)^{\vec{x}}}{\Gamma^{\vec{x}}+\Delta^{\vec{x}}}\,\mathsf{m} \quad \text{with } (I_1\wedge\cdots\wedge I_n)+\!\!+(J_1\wedge\cdots\wedge J_n) \triangleq (I_1+J_1)\wedge\cdots\wedge(I_n+J_n)$$

An example is the derivation (5), for the term $(\lambda z.z\langle z\rangle)\langle x,x,y\rangle$ of Example 10.

$$\dfrac{\dfrac{\top}{((A{\to}B)+A){\to}\dfrac{\dfrac{((A{\to}B)+A)^z}{(A{\to}B)^z\wedge A^z}\,@}{B}}\,\lambda \;\wedge\; \dfrac{\dfrac{\dfrac{((A{\to}B)+A)^x}{((A{\to}B)+A+\langle\rangle)^x}\leq \;\wedge\; \dfrac{A^y}{(\langle\rangle+\langle\rangle+A)^y}\leq}{\dfrac{(A{\to}B)^x\wedge\dfrac{\langle\rangle^y}{\top}\,\triangle}{A{\to}B}=\;+\;\dfrac{A^x\wedge\dfrac{\langle\rangle^y}{\top}\,\triangle}{A}=\;+\;\dfrac{\dfrac{\langle\rangle^x}{\top}\,\triangle\wedge A^y}{A}=}\,\mathsf{m}}{(A{\to}B)+A}\leq}{B}\,@ \tag{5}$$

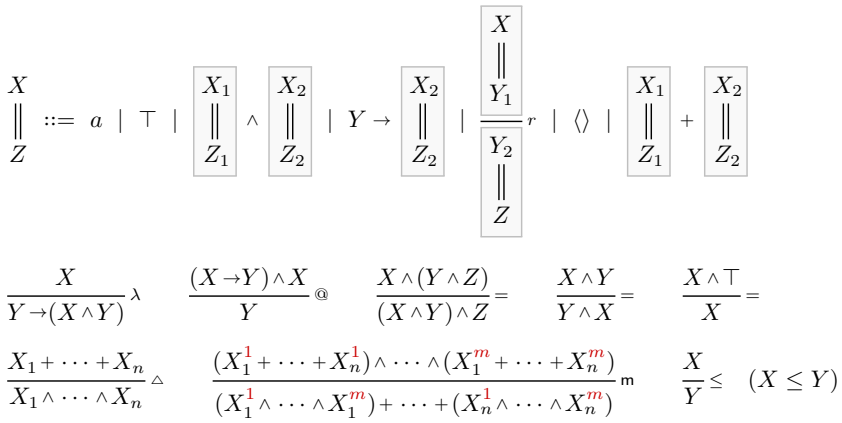
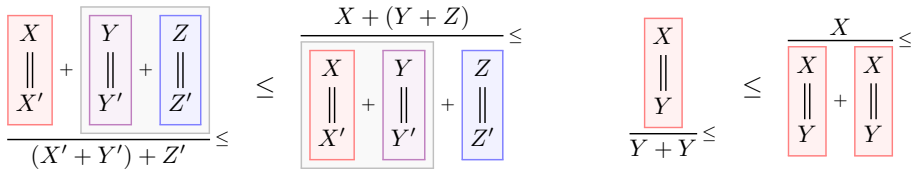


Figure 4 An open-deduction system for quantitative types

The inequality for collection terms $\tau \leq \sigma$, which expands during reduction, is captured in derivations by a permutation across the $\leq$ inference rule. We give *associativity* and *duplicability* as an example:



Typing the collection calculus imposes several equivalences on the quantitative open-deduction system. In Figure 6 we give the equivalences due to interaction of contraction and medial, and medial with itself: the first two are the splitting and associativity of contraction ($\triangle\triangle$) and ($\triangle\!\!\triangle$), adjusted from the simple type system for the SC; the latter two ($\mathsf{m}\,\mathsf{m}$) and ($\mathsf{m}\,\mathsf{m}$) are associativity laws for the medial. We equate the 1×1-medial and the unary contraction with the identity, illustrated below left, though we may choose to leave these rules as "markers" in the derivation for the term constructs $\langle s \rangle$, respectively $t[x \leftarrow \tau]$ where $|t|_x = 1$, to eliminate both with the rewrite rule for $t[x \leftarrow \langle s \rangle]$, as in Figure 7. We assume the following two *naturality* laws, for abstraction and medial.

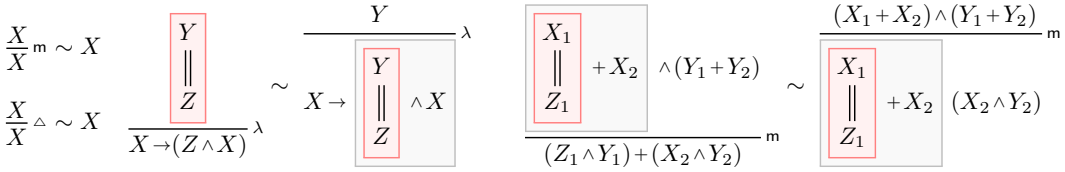


Figure 7 gives typing derivations for the reduction rules of the collection calculus, witnessing the following proposition.

▶ **Proposition 16** (Subject reduction). *The quantitative type system for the collection calculus satisfies subject reduction.*

Typing restricts reduction in the collection calculus: the terms in a collection may have different types, and only those with the same type as a given variable may be substituted for

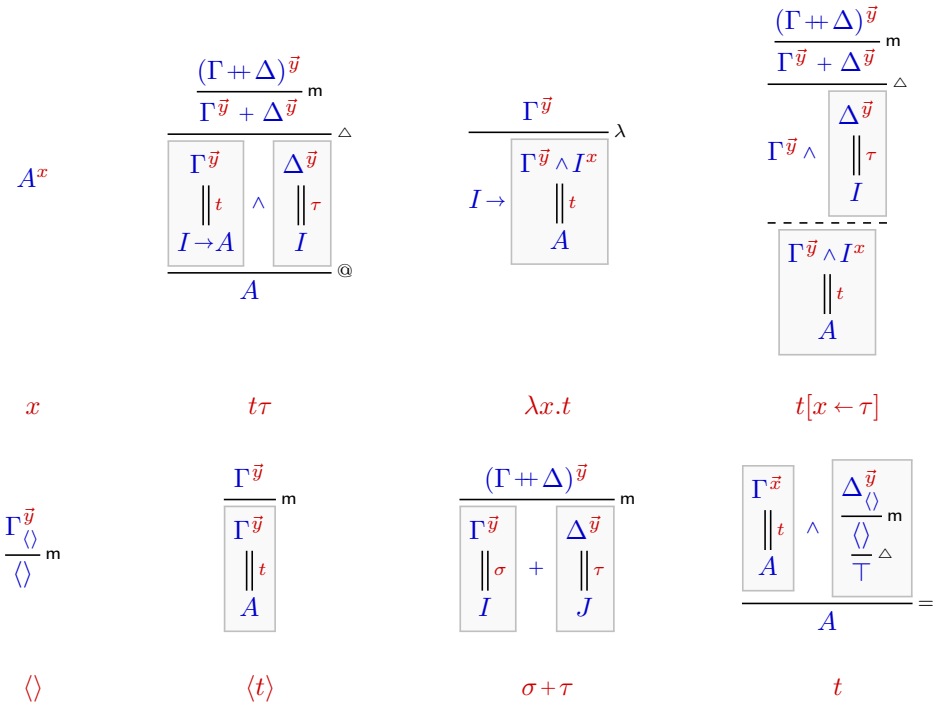
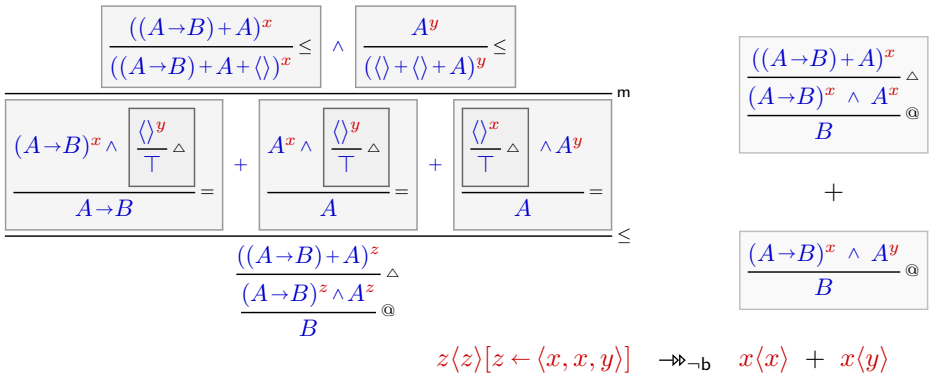


Figure 5 *Typing the collection calculus in open deduction.* In the constructions $t\tau$ and $t[x\leftarrow\tau]$ the contexts of both derivations are expanded to cover the same context variables $\vec{y}$, using the bottom-right figure. For non-empty collection derivations the context is expanded for every term, while empty collections are given for any context variables $\vec{y}$.

it. The example (6) gives the typed $\neg$b-reduction (omitting the first b-step) of Example 10 with *duplicability* and *redundancy*. The typed reduction results in only two of the four summands of the untyped reduction: the remaining two, $y\langle x\rangle$ and $y\langle y\rangle$, are not well-typed with respect to this reduction, since the derivation for $z\langle z\rangle[z\leftarrow\langle x,x,y\rangle]$ does not assign y the correct type to appear in function position.



(6)

▶ **Theorem 17.** *A typed CC-term is strongly normalizing.*

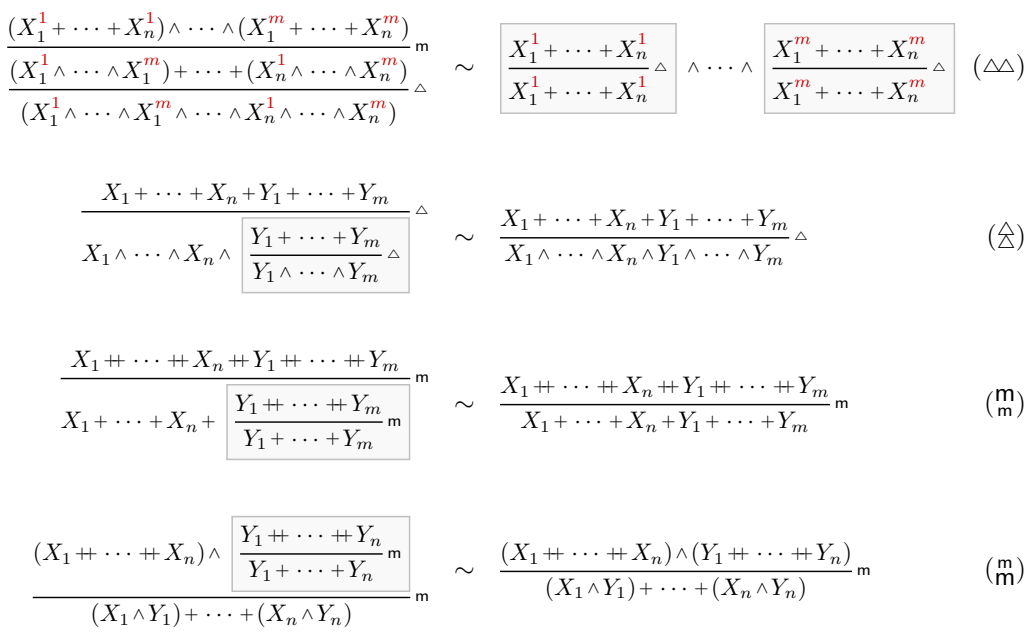


Figure 6 Equivalences for contraction and medial

6 Intersection types

Resource calculi aim to provide a notion of *approximation* of λ-terms, as an alternative to that given by Böhm trees. The purpose of collections in such calculi is to approximate arbitrary duplication (of a function argument) by a pre-determined, finite amount of duplication. Figure 1 in the introduction demonstrates how in deep inference, this pre-determined duplication can be implemented by rewriting. The difference with Böhm trees is exactly that the latter do not separate duplication from beta-reduction, where resource calculi do.

In our case, a CC-term may approximate an SC-term, which we formalize below by the relation $t \succ s$. Via this approximation the quantitative type system of the CC becomes an intersection type system for the SC, similar to the approach of Kfoury [30].

▶ **Definition 18.** *The* uniformity *law requires CC-terms to be* uniform*, as follows. A collection term t* flattens *to a structural λ-term s, and s* expands *to t, by the inductive relation $t \succ s$:*

$$\frac{}{x \succ x} \quad \frac{t \succ s \quad \tau \succ u}{t\tau \succ su} \quad \frac{t \succ s}{\lambda x.t \succ \lambda x.s} \quad \frac{t \succ s \quad \tau \succ u}{t[x \leftarrow \tau] \succ s[x \leftarrow u]} \quad \frac{}{\langle\rangle \succ u} \quad \frac{t \succ u}{\langle t \rangle \succ u} \quad \frac{\sigma \succ u \quad \tau \succ u}{\sigma + \tau \succ u}$$

A uniform *collection term t is one equipped with a flattening s, written as the pair $t \succ s$.*

Subterms of a uniform term receive their annotation inductively. Observe that s in $t \succ s$ is uniquely defined except at subterms of the form $\langle\rangle$ in t. During reduction, collections must be kept uniform: every term t_i in a collection $\langle t_1, \dots, t_n \rangle \succ t$ must be reduced simultaneously, along a reduction $t \to_{\mathsf{sc}} s$. We need the following, which is an immediate induction.

▶ **Proposition 19.** *If $t \succ s \to_{\mathsf{sc}} s'$ then $t \twoheadrightarrow_{\mathsf{cc}} t' \succ s'$ for some CC-term t'.*

▶ **Definition 20.** *A* uniform reduction step $(t \succ s) \to (t' \succ s')$ *is a reduction step $s \to s'$ lifted to a corresponding reduction $t \twoheadrightarrow t'$ along the inductive definition of $\succ$.*

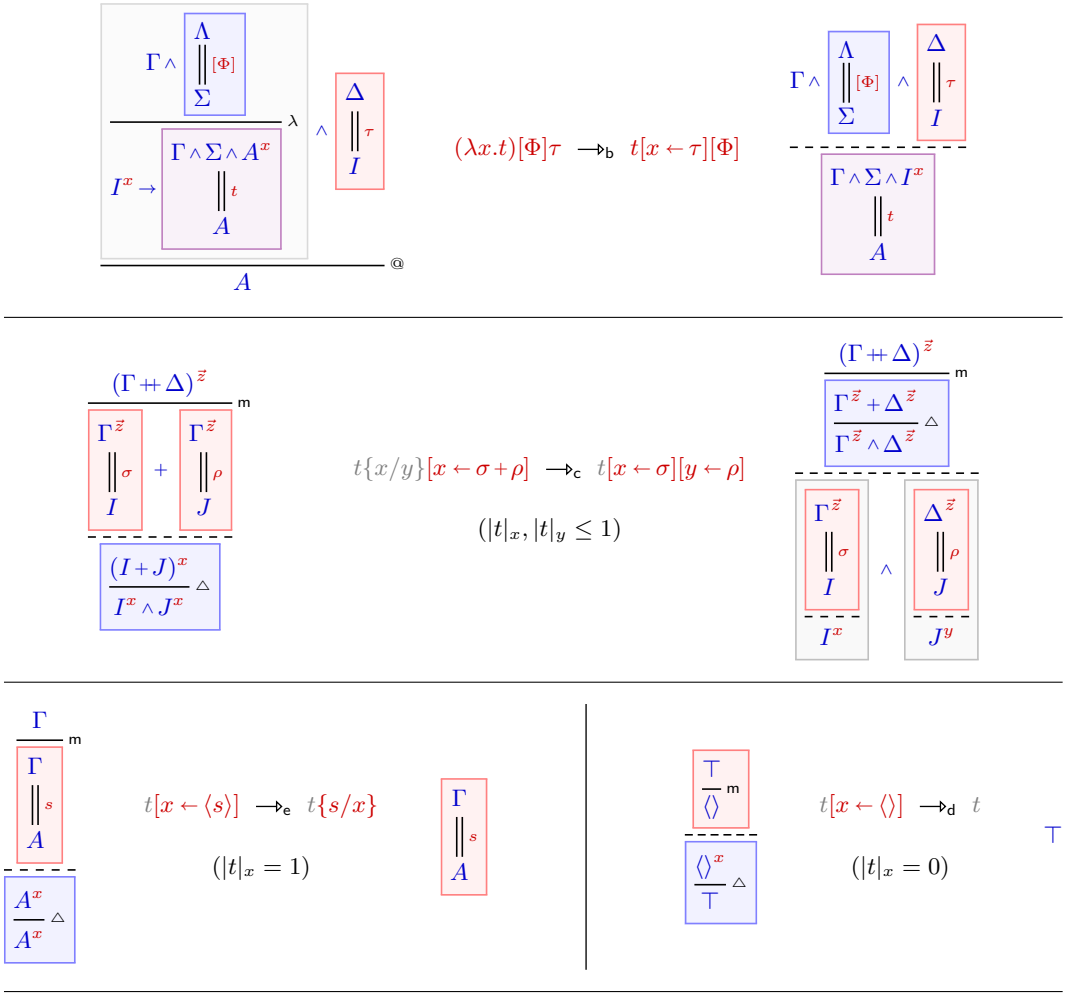


Figure 7 Subject reduction for the typed collection calculus

A derivation for a uniform CC-term $u \succ t$ is an intersection type derivation for the SC-term t. With *idempotence*, we have *idempotent intersection types*; without, *non-idempotent intersection types*. Both characterize weak normalization, since a collection $\langle\rangle \succ t$ may be equipped with a non-normalizing SC-term t, and nevertheless typed by the empty type $\langle\rangle$. To capture *strong* normalization, we adjust the type system to ask a typing witness for t.

▶ **Definition 21.** *The* strength *law introduces an inference rule* s *and replaces the typing law for* $\langle\rangle \succ t$ *by:*

$$\frac{A}{\langle\rangle}\,\mathsf{s} \qquad \langle\rangle \succ t : \quad \frac{\Gamma^{\vec{y}}}{\langle\rangle}\,\mathsf{m}\ \ \Gamma^{\vec{y}} \Vert_{r \succ t} A \,\mathsf{s} \qquad \text{for some uniform CC-term } r \succ t$$

▶ **Theorem 22.** *A structural* λ*-term* t *is weakly [strongly] normalizing if and only if there is a typed, [strong,] uniform CC-term* $u \succ t$*.*

7 Discussion and future work

Type uniformity and simple types Analogous to term uniformity, a *type uniformity* law may require that a quantitative type comes equipped with a flattening onto a *simple type*:

$$\frac{}{a \succ a} \qquad \frac{I \succ B \quad A \succ C}{I \to A \succ B \to C} \qquad \frac{}{\langle\rangle \succ A} \qquad \frac{I \succ A \quad J \succ A}{I + J \succ A} \qquad \frac{}{\top \succ \top} \qquad \frac{\Gamma \succ \Sigma \quad \Delta \succ \Lambda}{\Gamma \wedge \Delta \succ \Sigma \wedge \Lambda}$$

Without further algebraic laws, *type uniformity* gives a quantitative version of simple types. With also *redundancy* and *duplicability* quantitative types collapse to simple types, as a collection and its flattening will behave equivalently. With *type uniformity* but not *term uniformity*, opposite to intersection types, we obtain a typed *non-deterministic* calculus, where a collection $\langle t_1, \dots, t_n \rangle$ of type $I \succ A$ represents a non-deterministic choice over terms t_i of type A. Exploring this connection more deeply is future work.

Relaxing associativity, unitality, symmetry In the quantitative type system, the laws of *associativity*, *unitality*, and *symmetry* apply to collections, and separately to the conjunction, through the corresponding proof rules. These laws can safely be relaxed, though for that to be meaningful, they must be relaxed for the conjunction as well as for collections.

For the collections of the CC, these laws can likewise be relaxed straightforwardly. However, corresponding to the conjunction in the type system is the variable policy of the CC, where *associativity*, *unitality*, and *symmetry* are implicit. Technically, the free variables of a term form a *set*, though since their number of occurrences is significant—in particular, it drives the reduction rules—they morally form a *multiset*. To relax these laws, then, the calculus must be reformulated. Consider the following *linear* variant of the structural λ-calculus, where variables occur once but abstractions and closures bind a vector $\vec{x} = x_1 \dots x_n$ of variables:

$$t \ ::= \ x \ \mid \ t\tau \ \mid \ \lambda\vec{x}.t \ \mid \ t[\vec{x} \leftarrow \tau]$$

Symmetry of conjunction becomes explicit in the order of variables in a vector $\vec{x}$. Relaxing symmetry of $(+)$ but not $(\wedge)$ yields a linear λ-calculus for intuitionistic multiplicative linear logic as in [8], where $(\to)$ behaves as $(\multimap)$, and $(\wedge)$ and $(+)$ together behave as $(\otimes)$ (strictly, also the distinction between terms and collection terms should be collapsed). Symmetry of conjunction is relaxed by imposing that variables are bound in the same order as they occur; i.e. the free variables of a term must become a vector, and binding restricted accordingly:

$$\begin{aligned}
\mathsf{fv}(x) &= x \\
\mathsf{fv}(t\tau) &= \vec{x}\vec{y} && \text{where } \mathsf{fv}(t) = \vec{x},\ \mathsf{fv}(\tau) = \vec{y} \\
\mathsf{fv}(\lambda\vec{x}.t) &= \vec{y} && \text{where } \mathsf{fv}(t) = \vec{y}\vec{x} \\
\mathsf{fv}(t[\vec{y}_1 \leftarrow \tau]) &= \vec{x}\vec{y}_2\vec{z} && \text{where } \mathsf{fv}(t) = \vec{x}\vec{y}_1\vec{z},\ \mathsf{fv}(\tau) = \vec{y}_2\ .
\end{aligned}$$

Such a construction significantly reduces the expressivity of the calculus. However, it is possible that such non-commutativity can model aspects of sequential (imperative) computation; investigating this is future work. Going further, *associativity* or *unitality* can be relaxed by replacing vectors $\vec{x}$ with collections over variables, akin to *pattern-matching*, though the limited expressiveness casts doubt on how useful this would be.

Further algebraic laws The present exposition restricts itself to a sample of common algebraic laws. It is clear that there is potentially a large range of laws that would fit within the current framework, as long as the fundamental proof rules $\triangle$, m, $\leq$ are unimpeded. Establishing this range is future work. In a similar direction, connections with the exponentials of linear logic, and their *light* versions [24], are also of interest.

References

1 Beniamino Accattoli. An abstract factorization theorem for explicit substitutions. In *23rd International Conference on Rewriting Techniques and Applications (RTA'12) , RTA 2012, May 28 - June 2, 2012, Nagoya, Japan*, volume 15 of *LIPIcs*, pages 6–21. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2012. doi:10.4230/LIPIcs.RTA.2012.6.

2 Beniamino Accattoli, Stéphane Graham-Lengrand, and Delia Kesner. Tight typings and split bounds, fully developed. *Journal of Functional Programming*, 30:e14, 2020. doi:10.1017/S095679682000012X.

3 Beniamino Accattoli and Delia Kesner. The structural lambda-calculus. In *International Workshop on Computer Science Logic (CSL)*, 2010.

4 Beniamino Accattoli and Delia Kesner. Preservation of strong normalisation modulo permutations for the structural lambda-calculus. *Logical Methods in Computer Science*, 8(1), 2012.

5 Juan P. Aguilera and Matthias Baaz. Unsound inferences make proofs shorter. *Journal of Symbolic Logic*, 84(1):102–122, 2019.

6 Andrea Aler Tubella and Alessio Guglielmi. Subatomic proof systems: Splittable systems. *ACM Transactions on Computational Logic (TOCL)*, 19(1:5):1–33, 2018.

7 Hendrik Pieter Barendregt, Mario Coppo, and Mariangiola Dezani-Ciancaglini. A filter lambda model and the completeness of type assignment. *The Journal of Symbolic Logic*, 48(4):931–940, 1983.

8 Nick Benton, Gavin Bierman, Valeria de Paiva, and Martin Hyland. A term calculus for intuitionistic linear logic. In *International Conference on Typed Lambda Calculi and Applications*, pages 75–90, 1993.

9 Gérard Boudol. The lambda-calculus with multiplicities. In *International Conference on Concurrency Theory (CONCUR)*, 1993.

10 Kai Brünnler and Richard McKinley. An algorithmic interpretation of a deep inference system. In *nternational Conference on Logic for Programming Artificial Intelligence and Reasoning (LPAR)*, pages 482–496, 2008.

11 Kai Brünnler and Alwen Tiu. A local system for classical logic. In *8th International Conference on Logic for Programming Artificial Intelligence and Reasoning (LPAR)*, volume 2250 of *Lecture Notes in Computer Science (LNCS)*, pages 347–361, 2001.

12 Paola Bruscoli, Alessio Guglielmi, Tom Gundersen, and Michel Parigot. Quasipolynomial normalisation in deep inference via atomic flows. *Logical Methods in Computer Science*, 12(2), 2016.

13 Antonio Bucciarelli and Thomas Ehrhard. On phase semantics and denotational semantics: the exponentials. *Annals of Pure and Applied Logic*, 109(3):205–241, 2001. doi:10.1016/S0168-0072(00)00056-7.

14 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.

15 Mario Coppo and Mariangiola Dezani-Ciancaglini. A new type assignment for lambda-terms. *Archiv für mathematische Logik und Grundlagenforschung*, 19(1):139–156, 1978.

16 Mario Coppo and Mariangiola Dezani-Ciancaglini. An extension of the basic functionality theory for the λ-calculus. *Notre Dame Journal of Formal Logic*, 21(4):685–693, 1980.

17 Ugo Dal Lago, Giulio Guerrieri, and Willem Heijltjes. Decomposing probabilistic lambda-calculi. In *Foundations of Software Science and Computation Structures - 23rd International Conference, FOSSACS 2020*, volume 12077 of *Lecture Notes in Computer Science*, pages 136–156. Springer, 2020. doi:10.1007/978-3-030-45231-5_8.

18 Daniel de Carvalho. The relational model is injective for multiplicative exponential linear logic. In *25th EACSL Annual Conference on Computer Science Logic, CSL 2016, August 29 - September 1, 2016, Marseille, France*, pages 41:1–41:19, 2016. URL: https://doi.org/10.4230/LIPIcs.CSL.2016.41, doi:10.4230/LIPIcs.CSL.2016.41.

19 Daniel de Carvalho. Execution time of λ-terms via denotational semantics and intersection types. *Mathematical Structures in Computer Science*, 28(7):1169–1203, 2018. doi:10.1017/S0960129516000396.

20 Thomas Ehrhard. Non-idempotent intersection types in logical form. In Jean Goubault-Larrecq and Barbara König, editors, *Foundations of Software Science and Computation Structures, FoSSaCS 2020*, volume 12077 of *LNCS*, pages 198–216, Cham, 2020. Springer International Publishing.

21 Thomas Ehrhard and Laurent Regnier. The differential lambda-calculus. *Theoretical Computer Science*, 309(1–3):1–41, 2003.

22 Thomas Ehrhard and Laurent Regnier. Uniformity and the Taylor expansion of ordinary lambda-terms. *Theoretical Computer Science*, 403:347–372, 2008.

23 Philippa Gardner. Discovering needed reductions using type theory. In *Theoretical Aspects of Computer Software, International Conference TACS '94, Sendai, Japan, April 19-22, 1994, Proceedings*, volume 789 of *Lecture Notes in Computer Science*, pages 555–574. Springer, 1994. URL: https://doi.org/10.1007/3-540-57887-0_115, doi:10.1007/3-540-57887-0_115.

24 Jean-Yves Girard. Light linear logic. *Information and Computation*, 143(2):175–204, 1998.

25 Alessio Guglielmi, Tom Gundersen, and Michel Parigot. A proof calculus which reduces syntactic bureaucracy. In *Rewriting Techniques and Applications (RTA)*, pages 135–150, 2010.

26 Tom Gundersen, Willem Heijltjes, and Michel Parigot. Atomic lambda-calculus: a typed lambda-calculus with explicit sharing. In *28th IEEE Symposium on Logic in Computer Science LICS*, pages 311–320, 2013.

27 Willem Heijltjes and Joe Paulus. Deep-inference intersection types. Extended abstract, presented at the workshop Twenty Years of Deep Inference (TYDI), Oxford, 2018. Available at http://willem.heijltj.es/pdf/2018-heijltjes-paulus.pdf, 2018.

28 Emil Jeřábek. Proof complexity of the cut-free calculus of structures. *Journal of Logic and Computation*, 19(2):323–339, 2009.

29 Delia Kesner and Stéphane Lengrand. Resource operators for lambda-calculus. *Information and Computation*, 205(4):419–473, 2007.

30 A.J. Kfoury. A linearization of the lambda-calculus and consequences. *Journal of Logic and Computation*, 10(3):411–436, 2000.

31 Jim Laird, Giulio Manzonetto, Guy McCusker, and Michele Pagani. Weighted relational models of typed lambda-calculi. In *28th IEEE Symposium on Logic in Computer Science (LICS)*, pages 301–310, 2013.

32 Damiano Mazza, Luc Pellissier, and Pierre Vial. Polyadic approximations, fibrations and intersection types. *Proceedings of the ACM on Programming Languages*, 2(POPL), 2018.

33 C.-H. Luke Ong. Quantitative semantics of the lambda-calculus: Some generalisations of the relational model. In *32nd Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, 2017.

34 Luca Paolini, Mauro Piccolo, and Simona Ronchi Della Rocca. Essential and relational models. *Mathematical Structures in Computer Science*, 27(5):626–650, 2017. doi:10.1017/S0960129515000316.

35 Elaine Pimentel, Simona Ronchi Della Rocca, and Luca Roversi. Intersection Types from a proof-theoretic perspective. *Fundamenta Informaticae*, 121(1-4):253–274, 2012.

36 G. Pottinger. *To H. B. Curry: essays on combinatory logic, lambda calculus and formalism*, chapter A type assignment for the strongly normalizable λ-terms, pages 561–577. Academic Press, London, 1980.

37 David Sherratt, Willem Heijltjes, Tom Gundersen, and Michel Parigot. Spinal atomic lambda-calculus. In Jean Goubault-Larrecq and Barbara König, editors, *Foundations of Software Science and Computation Structures - 23rd International Conference, FOSSACS 2020*, volume 12077 of *LNCS*, pages 582–601, Cham, 2020. Springer International Publishing. doi:10.1007/978-3-030-45231-5_30.

38 Alwen Tiu. A local system for intuitionistic logic. In *International Conference on Logic for Programming Artificial Intelligence and Reasoning (LPAR)*, pages 242–256, 2006.

39 Alwen Fernanto Tiu. A system of interaction and structure II: The need for deep inference. *Logical Methods in Computer Science*, 2(2):4:1–24, 2006.

Appendix

A Encoding non-idempotent intersection types and resource calculi

Resource λ-calculus The *λ-calculus with multiplicities* by Boudol [9] features two-sorted collections P, Q with both non-duplicable and duplicable elements, the latter indicated M^∞:

$$M, N \ ::= \ x \ \mid \ NP \ \mid \ \lambda x.N \ \mid \ N[P/x] \qquad P, Q \ ::= \ 1 \ \mid \ M \ \mid \ (P|P) \ \mid \ M^\infty$$

It employs *weak head reduction*, with β-reduction occurring in head context H and closures evaluated by substituting into head variables $H\{x\}$: (borrowing the CC-notation $[\Phi]$)

$$H \ ::= \ \{\} \ \mid \ HP \ \mid \ H[P/x] \qquad \begin{array}{l} H\{(\lambda x.N)[\Phi]P\} \ \to_{\mathsf{b}} \ H\{N[P/x][\Phi]\} \\ H\{x\}[(N|P)/x] \ \to_{\mathsf{s}} \ H\{N\}[P/x] \end{array}$$

Note that s-reduction is non-deterministic. The two-sorted collections can be imported into the CC ad-hoc by admitting collections t^∞ and the law $t^\infty \leq \langle t \rangle + t^\infty$. b-Reduction is as in the CC, and informally translating N to t, P to τ, and H to h, s-reduction is simulated by:

$$h\{x\}[x \leftarrow \langle t \rangle + \tau] \ \to_{\mathsf{c}} \ h\{y\}[y \leftarrow \langle t \rangle][x \leftarrow \tau] \ \to_{\mathsf{e}} \ h\{t\}[x \leftarrow \tau]$$

Non-idempotent intersection types Kfoury in [30] gives non-idempotent intersection types via a resource calculus with *uniformity*. Resource terms are given below, where a term N is *well-formed* if it flattens to a regular λ-term $|N|$. β-Reduction on N is defined along $|N|$.

$$M, N, P \ ::= \ x \ \mid \ \lambda x.N \ \mid \ N.\, P_1 \wedge \cdots \wedge P_n \quad (n \geq 1)$$

$$\frac{}{|x| = x} \qquad \frac{|N| = M}{|\lambda x.N| = \lambda x.M} \qquad \frac{|N| = M \quad \{|P_i| = Q\}_{1 \leq i \leq n}}{|N.\, P_1 \wedge \cdots \wedge P_n| = M\, Q}$$

A system of simple types $\boldsymbol{\lambda}^\wedge$ for the resource calculus, given below, then generates a system of non-idempotent intersection types $\boldsymbol{\lambda}$ for the regular λ-calculus by flattening the terms in each typing rule. We use our notational conventions for types, and define the $\cup$ operator by letting $(\Gamma^{\vec{x}} \wedge \Delta^{\vec{y}}) \cup (\Lambda^{\vec{y}} \wedge \Sigma^{\vec{z}}) = \Gamma^{\vec{x}} \wedge (\Delta +\!\!+ \Lambda)^{\vec{y}} \wedge \Sigma^{\vec{z}}$ if $\vec{x}$ and $\vec{z}$ share no variables.

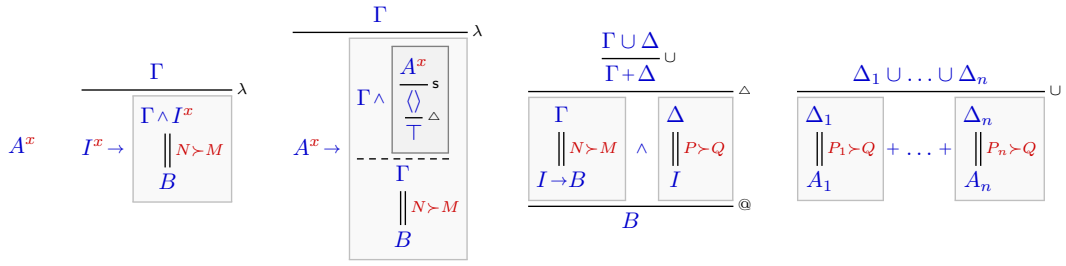
$$\frac{}{x : A \vdash x : A} \qquad \frac{\Gamma, x : I \vdash N : B}{\Gamma \vdash \lambda x.N : I \to B}\, {}^{I \neq \langle\rangle} \qquad \frac{\Gamma \vdash N : B}{\Gamma \vdash \lambda x.N : A \to B} \qquad \frac{\Gamma \vdash N : (A_1 + \ldots + A_n) \to B \quad \{\Delta_i \vdash P_i : A_i\}_{1 \leq i \leq n}}{\Gamma \cup \Delta_1 \cup \cdots \cup \Delta_n \vdash N.\, P_1 \wedge \cdots \wedge P_n : B}$$

Systems $\boldsymbol{\lambda}^\wedge$ and $\boldsymbol{\lambda}$ are effectively a restriction of the typed collection calculus to terms without closures, with the laws of *uniformity* and *strength*. Strictly, also *symmetry* is dropped, but there is no loss of expressiveness because the conjunction remains symmetric and because of *uniformity*. We introduce an admissible proof rule for the $\cup$ operator:

$$\frac{(\Gamma^{\vec{x}} \wedge \Delta^{\vec{y}}) \cup (\Lambda^{\vec{y}} \wedge \Sigma^{\vec{z}})}{(\Gamma^{\vec{x}} \wedge \Delta^{\vec{y}}) + (\Lambda^{\vec{y}} \wedge \Sigma^{\vec{z}})}\, \cup \quad = \quad \frac{\Gamma^{\vec{x}} \wedge (\Delta +\!\!+ \Lambda)^{\vec{y}} \wedge \Sigma^{\vec{z}}}{\Gamma^{\vec{x}} \wedge \Delta^{\vec{y}} \wedge \dfrac{\Sigma^{\vec{z}}_{\langle\rangle}}{\dfrac{\langle\rangle}{\top}\,\triangle}\,\mathsf{m} \ + \ \dfrac{\Gamma^{\vec{x}}_{\langle\rangle}}{\dfrac{\langle\rangle}{\top}\,\triangle}\,\mathsf{m} \wedge \Lambda^{\vec{y}} \wedge \Sigma^{\vec{z}}}\,\mathsf{m}$$

Systems $\boldsymbol{\lambda}^\wedge$ and $\boldsymbol{\lambda}$ are then encoded as follows, where $N \succ M$ is $|N| = M$, and where the constructions for application and collections are kept separate, with $\Delta = \Delta_1 \cup \cdots \cup \Delta_n$,

$P = P_1 \wedge \cdots \wedge P_n$, and $I = A_1 + \cdots + A_n$.



B Omitted proofs and lemmas in Section 2

▶ Remark 23 (Free variable and translation). For every SC-term t, $\mathsf{fv}(t^\bullet) \subseteq \mathsf{fv}(t)$. The proof is by straightforward induction on t.

▶ **Lemma 24** (Substitution). *For any sharing terms t and u, we have $(t\{u/x\})^\bullet = t^\bullet\{u^\bullet/x\}$.*

Proof. By straightforward induction on t. The only interesting case is the one with sharing: if $t = r[y \leftarrow s]$ (we can suppose without loss of generality that $y \notin \mathsf{fv}(u) \cup \{x\}$), then $t\{u/x\} = r\{u/x\}\{u/x\}[y \leftarrow s\{u/x\}]$ and $t^\bullet = r^\bullet\{s^\bullet/y\}$; by induction hypothesis, $r' = (r\{u/x\})^\bullet = r^\bullet\{u^\bullet/x\}$ and $s' = (s\{u/x\})^\bullet = s^\bullet\{u^\bullet/x\} = s'$, hence

$$(t\{u/x\})^\bullet = r'^\bullet\{s'^\bullet/y\} = r^\bullet\{u^\bullet/x\}\{s^\bullet\{u^\bullet/x\}/y\} = r^\bullet\{s^\bullet/y\}\{u^\bullet/x\} = t^\bullet\{u^\bullet/x\}. \qquad \blacktriangleleft$$

▶ **Proposition 3** (Simulations). *Let t be a SC-term and s be a λ-term.*

1. From SC to λ-calculus*: If $t \to_{\mathsf{b}} t'$ then $t^\bullet \to_\beta t'^\bullet$; if $t \to_{\neg\mathsf{b}} t'$ then $t^\bullet = t'^\bullet$.*
2. From λ-calculus to SC*: If $s \to_\beta s'$ then $s \to_{\mathsf{b}} \twoheadrightarrow_{\neg\mathsf{b}} s'$.*

Proof. 1. Both proofs are by induction on the SC-term t. We omit some cases that easily follows from the induction hypothesis.

- Let us prove that if $t \to_{\mathsf{b}} t'$, then $t^\bullet \twoheadrightarrow_\beta t'^\bullet$. Cases of interest:
  - If $t = (\lambda y.s)[x_1 \leftarrow r_1]\dots[x_n \leftarrow r_n]u \to_{\mathsf{b}} s[y \leftarrow u][x_1 \leftarrow r_1]\dots[x_n \leftarrow r_n] = t'$, then we can suppose without loss of generality that $y \notin \bigcup_{i=1}^n \mathsf{fv}(r_i) \cup \{x\}$ and so

$$\begin{aligned} t^\bullet = (\lambda y.s^\bullet)\{r_1^\bullet/x_1\}\dots\{r_n^\bullet/x_n\}u^\bullet &= (\lambda y.s^\bullet\{r_1^\bullet/x_1\}\dots\{r_n^\bullet/x_n\})u^\bullet \\ \to_\beta s^\bullet\{r_1^\bullet/x_1\}\dots\{r_n^\bullet/x_n\}\{u^\bullet/y\} &= s^\bullet\{u^\bullet/y\}\{r_1^\bullet/x_1\}\dots\{r_n^\bullet/x_n\} \\ &= (s[y \leftarrow u])^\bullet\{r_1^\bullet/x_1\}\dots\{r_n^\bullet/x_n\} = t'^\bullet \end{aligned}$$

    where the second to last equality holds because of substitution lemma (Lemma 24).
  - If $t = u[x \leftarrow s] \to_{\mathsf{b}} u[x \leftarrow s'] = t'$ with $s \to_{\mathsf{b}} s'$ then, by induction hypothesis, $s^\bullet \to_\beta^* s'^\bullet$ and so $t^\bullet = u^\bullet\{s^\bullet/x\} \to_\beta^* u^\bullet\{s'^\bullet/x\} = t'^\bullet$.

- Let us prove that if $t \to_{\neg\mathsf{b}} t'$, then $t^\bullet = t'^\bullet$. Cases of interest:
  - *Copy*: if $t = s\{x/y\}[x \leftarrow u] \to_{\mathsf{c}} s[x \leftarrow u][y \leftarrow u] = t'$, we can suppose without loss of generality that $y \notin \mathsf{fv}(u)$. By Lemma 24, $(s\{x/y\})^\bullet = s^\bullet\{x/y\}$ and $(s[x \leftarrow u])^\bullet = s^\bullet\{u^\bullet/x\} = (s\{u/x\})^\bullet$. Thus,

$$t^\bullet = (s\{x/y\})^\bullet\{u^\bullet/x\} = s^\bullet\{x/y\}\{u^\bullet/x\} = s^\bullet\{u^\bullet/x\}\{u^\bullet/y\} = (s[x \leftarrow u])^\bullet\{u^\bullet/y\} = t'^\bullet.$$

  - *Delete*: if $t = s[x \leftarrow u] \to_{\mathsf{d}} s = t'$ then $x \notin \mathsf{fv}(s)$ and hence $x \notin \mathsf{fv}(s^\bullet)$ by Remark 23, so $t^\bullet = s^\bullet\{u^\bullet/x\} = s^\bullet = t'^\bullet$.

- *Evaluate*: if $t = u[x \leftarrow s] \rightarrow_{\mathsf{e}} u\{s/x\} = t'$ then, by substitution lemma (Lemma 24),

$$t^\bullet = s^\bullet\{u^\bullet/x\} = (s\{u/x\})^\bullet = t'^\bullet.$$

2. See [4, proof of Lemma 2.4] and apply Proposition 4.1 below (proved independently). ◀

▶ **Proposition 4** (Collated results from [4]). *The SC has the following key properties.*
1. *The normal forms of $\rightarrow_{\neg\mathsf{b}}$ are exactly the λ-terms.*
2. *The normal forms of $\rightarrow_{\mathsf{sc}}$ are exactly the β-normal λ-terms.*
3. *For any SC-term t, one has $t \twoheadrightarrow_{\neg\mathsf{b}} t^\bullet$; in particular, $t = t^\bullet$ for any λ-term.*
4. *The relations $\rightarrow_{\mathsf{b}}$, $\rightarrow_{\neg\mathsf{b}}$, and $\rightarrow_{\mathsf{sc}}$ are confluent; $\rightarrow_{\mathsf{b}}$ and $\rightarrow_{\neg\mathsf{b}}$ are strongly normalizing.*
5. *Preservation of strong normalization: if a λ-term t has an infinite $\rightarrow_{\mathsf{sc}}$-reduction, then it has an infinite $\rightarrow_{\beta}$-reduction.*

Proof. 1. Clearly, every λ-term is normal for $\rightarrow_{\neg\mathsf{b}}$ because there is no sharing. Conversely, if t is a sharing term that is normal for $\rightarrow_{\neg\mathsf{b}}$ then there are no context C and no sharing terms s and u such that $t = C\langle s[x \leftarrow u]\rangle$, otherwise if $|s|_x = 0$ then t would not be normal for $\rightarrow_{\mathsf{d}}$, if $|s|_x = 1$ then t would not be normal for $\rightarrow_{\mathsf{e}}$, if $|s|_x > 1$ then t would not be normal for $\rightarrow_{\mathsf{c}}$; therefore, t has no sharings and hence is a λ-term.
2. Since $\rightarrow_{\mathsf{sc}} = \rightarrow_{\mathsf{b}} \cup \rightarrow_{\neg\mathsf{b}}$ and in SC the normal forms of $\rightarrow_{\neg\mathsf{b}}$ are exactly the λ-terms, it is enough to observe that a λ-term t is normal for $\rightarrow_{\mathsf{b}}$ if and only if there are no λ-context C and λ-terms s and u such that $t = C\langle(\lambda x.s)u\rangle$, which amounts to say that t is β-normal.
3. First, if t is a λ-term then $t^\bullet = t$ since there are no sharings in t. Now, let t be a SC-term, with $t \twoheadrightarrow_{\neg\mathsf{b}} s$ (such a s exists because $\rightarrow_{\neg\mathsf{b}}$ is strongly normalizing, [4, Lemma 2.10]); as s is a λ-term (Proposition 4.1), we have just shown that $s = s^\bullet$; by Proposition 3.1, $s^\bullet = t^\bullet$ and so $t \twoheadrightarrow_{\neg\mathsf{b}} t^\bullet$.
4. Strong normalization of $\rightarrow_{\mathsf{b}}$ is trivial (each step decreases the number of applications). Strong normalization of $\rightarrow_{\neg\mathsf{b}}$ is proved in [4, Lemma 2.10]
Accattoli and Kesner [3, 4] already proved confluence of $\rightarrow_{\mathsf{sc}}$ (using Tait–Martin-Löf's technique based on parallel reduction) and of $\rightarrow_{\neg\mathsf{b}}$ (via Newman's lemma). Here we present a simpler, modular and more informative proof, which relies on the confluence of β reduciton.
It is easy to check that $\rightarrow_{\mathsf{b}}$ has the diamond property and hence is confluent.
Concerning confluence of $\rightarrow_{\neg\mathsf{b}}$, suppose $s \;{}_{\neg\mathsf{b}}\!\twoheadleftarrow t \twoheadrightarrow_{\neg\mathsf{b}} u$. By Proposition 4.3, $s \twoheadrightarrow_{\neg\mathsf{b}} s^\bullet$ and $r \twoheadrightarrow_{\neg\mathsf{b}} r^\bullet$. According to Proposition 3.1, $s^\bullet = t^\bullet = r^\bullet$.
Concerning confluence of $\rightarrow_{\mathsf{sc}}$, suppose $s \;{}_{\mathsf{sc}}\!\twoheadleftarrow t \twoheadrightarrow_{\mathsf{sc}} u$. By Proposition 4.3, $s \twoheadrightarrow_{\neg\mathsf{b}} s^\bullet$ and $r \twoheadrightarrow_{\neg\mathsf{b}} r^\bullet$. According to Proposition 3.1, $s^\bullet \;{}_{\beta}\!\twoheadleftarrow t^\bullet \twoheadrightarrow_{\beta} r^\bullet$. By confluence of $\rightarrow_{\beta}$, $s^\bullet \twoheadrightarrow_{\beta} u \;{}_{\beta}\!\twoheadleftarrow r^\bullet$ and so $s \twoheadrightarrow_{\neg\mathsf{b}} s^\bullet \rightarrow_{\mathsf{b}}\twoheadrightarrow_{\neg\mathsf{b}} u \;{}_{\neg\mathsf{b}}\!\twoheadleftarrow{}_{\mathsf{b}}\!\leftarrow r^\bullet \;{}_{\neg\mathsf{b}}\!\twoheadleftarrow r$ by Proposition 3.2.
5. See [4, Lemma 3.5] ◀

C Omitted proofs and lemmas in Section 4

Let $|t|_x^{\mathrm{p}}$ be the maximal number of free occurrences of x that may appear in a ¬b-reduction sequence from the CC-term t. Formally:

$$|x|_x^{\mathrm{p}} = 1 \qquad |y|_x^{\mathrm{p}} = 0 \qquad |t\tau|_x^{\mathrm{p}} = |t|_x^{\mathrm{p}} + |\tau|_x^{\mathrm{p}} \qquad |\lambda y.t|_x^{\mathrm{p}} = |t|_x^{\mathrm{p}}$$

$$|t[y \leftarrow \tau]|_x^{\mathrm{p}} = |t|_x^{\mathrm{p}} + \max\{1, |t|_y^{\mathrm{p}}\} \cdot |\tau|_x^{\mathrm{p}} \qquad |\langle t_1, \ldots, t_n\rangle|_x^{\mathrm{p}} = \sum_{i=1}^{n} |t_i|_x^{\mathrm{p}}$$

▶ Remark 25. For any CC-term, $|t|_x^{\mathrm{p}} = 0$ if and only if $|t|_x = 0$.

▶ **Lemma 26.** *For any CC-term t, if $|t|_x, |t|_y \geq 1$, then $|t\{x/y\}|^{\mathrm{p}}_x = |t|^{\mathrm{p}}_x + |t|^{\mathrm{p}}_y$*

Proof. By induction on t. ◀

Inspired by [4], let us define the *size* $\|t\|$ of a CC-term t the following multiset of natural numbers (well-ordered by the multiset ordering):

$$\|x\| = [] \qquad \|\lambda x.t\| = \|t\| \qquad \|t\tau\| = \|t\| + \|\tau\|$$

$$\|t[y \leftarrow \tau]\| = \|t\| + [|t|^{\mathrm{p}}_x] + \max\{1, |t|^{\mathrm{p}}_x\} \cdot \|\tau\| \qquad \|\langle t_1, \dots, t_n\rangle\| = \sum_{i=1}^{n} \|t_i\|$$

▶ **Lemma 27.** *Without* duplicability, *$\|\tau\| \geq \|\sigma\|$ for any collection $\tau \geq \sigma$*

Proof. Trivial. ◀

▶ **Lemma 28.** *Let u be a CC-term.*

1. *If $|u|_x = 1$ then $\|u[x \leftarrow \tau]\| > \|u\{\langle s\rangle/x\}\|$ for any CC-term s such that $\tau \geq \langle s\rangle$.*
2. *$\|u\{x/y\}\| = \|u\|$.*

Proof. Both are by induction on the CC-term t. ◀

▶ **Proposition 13.** *$\neg\mathsf{b}$-Reduction of a CC-term t is strongly normalizing, and confluent in the following sense: if $t \twoheadrightarrow_{\neg\mathsf{b}} \sum_{s\in S} s$ and $t^\bullet = \sum_{r\in R} r$ then $S \subseteq R$.*

Proof. Let us prove that $t \to_{\neg\mathsf{b}} \sum_{i=1}^{n} t_i \ni t'$ implies $\|t\| > \|t'\|$, by induction on t. Cases:

- *Delete*: if $t = s[x \leftarrow \tau] \to_{\mathsf{d}} \sum_{\tau\leq\langle\rangle} s \ni t'$ with $|s|_x = 0$, then $t' = s$. By Remark 25, $|s|^{\mathrm{p}}_x = 0$. So, $\|t\| = \|s\| + [|s|^{\mathrm{p}}_x] + \max\{1, |s|^{\mathrm{p}}_x\} \cdot \|\tau\| = \|s\| + [|s|^{\mathrm{p}}_x] + \|\tau\| > \|s\| = \|t'\|$.
- *Substitution*: if $t = u[x \leftarrow \tau] \to_{\mathsf{e}} \sum_{\tau\leq\langle s\rangle} u\{s/x\} \ni t'$ with $|u|_x = 1$, then $t' = u\{s/x\}$. By Lemma 28.1, $\|t\| = \|u[x \leftarrow \tau]\| > \|u\{\langle s\rangle/x\}\| = t'$.
- *Copy*: if $t = s\{x/y\}[x \leftarrow \tau] \to_{\mathsf{c}} \sum_{\tau\leq\rho+\sigma} s[x \leftarrow \rho][y \leftarrow \sigma] \ni t'$ with $|s|_x, |s|_y \geq 1$, then $t' = s[x \leftarrow \rho][y \leftarrow \sigma]$.

$$\begin{aligned}
\|t\| &= \|s\{x/y\}\| + [|s\{x/y\}|^{\mathrm{p}}_x] + \max\{1, |s\{x/y\}|^{\mathrm{p}}_x\} \cdot \|\tau\| \\
&= \|s\| + [|s|^{\mathrm{p}}_x + |s|^{\mathrm{p}}_y] + \max\{1, |s\{x/y\}|^{\mathrm{p}}_x\} \cdot \|\tau\| && \text{L. 28.2 and 26} \\
&= \|s\| + [|s|^{\mathrm{p}}_x + |s|^{\mathrm{p}}_y] + |s\{x/y\}|^{\mathrm{p}}_x \cdot \|\tau\| && \text{Rmk. 25} \\
&= \|s\| + [|s|^{\mathrm{p}}_x + |s|^{\mathrm{p}}_y] + (|s|^{\mathrm{p}}_x + |s|^{\mathrm{p}}_y) \cdot \|\tau\| && \text{L. 26} \\
&> \|s\| + [|s|^{\mathrm{p}}_x] + |s|^{\mathrm{p}}_x \cdot \|\rho\| + [|s|^{\mathrm{p}}_y + |s|^{\mathrm{p}}_x \cdot |\rho|^{\mathrm{p}}_y] + (|s|^{\mathrm{p}}_y + |s|^{\mathrm{p}}_x \cdot |\rho|^{\mathrm{p}}_y) \cdot \|\sigma\| && \text{L. 27} \\
&= \|s\| + [|s|^{\mathrm{p}}_x] + |s|^{\mathrm{p}}_x \cdot \|\rho\| + [|s|^{\mathrm{p}}_y + \max\{1, |s|^{\mathrm{p}}_x\} \cdot |\rho|^{\mathrm{p}}_y] + (|s|^{\mathrm{p}}_y + |s|^{\mathrm{p}}_x \cdot |\rho|^{\mathrm{p}}_y) \cdot \|\sigma\| && \text{Rmk. 25} \\
&= \|s\| + [|s|^{\mathrm{p}}_x] + |s|^{\mathrm{p}}_x \cdot \|\rho\| + [|s[x \leftarrow \rho]|^{\mathrm{p}}_y] + (|s|^{\mathrm{p}}_y + \max\{1, |s|^{\mathrm{p}}_x\} \cdot |\rho|^{\mathrm{p}}_y) \cdot \|\sigma\| && \text{Rmk. 25} \\
&= \|s\| + [|s|^{\mathrm{p}}_x] + \max\{1, |s|^{\mathrm{p}}_x\} \cdot \|\rho\| + [|s[x \leftarrow \rho]|^{\mathrm{p}}_y] + \max\{1, |s[x \leftarrow \rho]|^{\mathrm{p}}_y\} \cdot \|\sigma\| && \text{Rmk. 25} \\
&= \|s[x \leftarrow \rho]\| + [|s[x \leftarrow \rho]|^{\mathrm{p}}_y] + \max\{1, |s[x \leftarrow \rho]|^{\mathrm{p}}_y\} \cdot \|\sigma\| \\
&= \|t'\|
\end{aligned}$$

- The other cases smoothly follow from the induction hypothesis. ◀

▶ **Proposition 15.** *The length of a (non-deterministic) reduction sequence $s \twoheadrightarrow_{\mathsf{cc}} t$ is:*

1. *without algebraic laws, exactly $|s| - |t|$;*
2. *with only* redundancy, *at most $|s| - |t|$;*
3. *with only* duplicability, *at least $|s| - |t|$.*

Proof. Recall the definition of reduction weight $|t|$ on page 10, where $x \notin \mathsf{fv}(r)$ but $x \in \mathsf{fv}(s)$. We consider each case; for the last two, observe that $|t|$ is always positive.

1. Each rewrite step reduces $|t|$ by exactly one: a step $(\lambda x.r)\tau \to_{\mathsf{b}} r[x \leftarrow \tau]$ where $x \in \mathsf{fv}(r)$ replaces an abstraction and application (weight zero) by a sharing (weight -1), and $(\lambda x.s)\tau \to_{\mathsf{b}} s[x \leftarrow \tau]$ where $x \in \mathsf{fv}(s)$ replaces weight 2 by weight 1; a c-step introduces a sharing of weight -1; an e-step removes a sharing (weight -1) and a variable (weight 1); and a d-step removes a weakening (weight 1).
2. By 1. above, and the observation that if $\tau \leq \sigma$ then $|\tau| \geq |\sigma|$.
3. By 1. above, and the observation that if $\tau \leq \sigma$ then $|\tau| \leq |\sigma|$. ◀

D Omitted proofs and lemmas in Section 5

▶ **Theorem 17.** *A typed CC-term is strongly normalizing.*

Proof sketch. We reduce the problem to typed normalization in a non-deterministic (or probabilistic) λ-calculus, which are known to terminate (see e.g. [17]).

First, the *copy* rewrite rule is *linear* for contractions on different types, since no idempotence law ($\geq$) applies. This corresponds to the following transformation on derivations:

$$\dfrac{\dfrac{\Gamma \quad \Delta}{A \quad B}}{\dfrac{(A+B)^x}{A \wedge B}\,\triangle} \quad \to_{\mathsf{c}} \quad \dfrac{\dfrac{\Gamma \wedge \Delta}{\Gamma \quad \Delta}\,\triangle}{A^x \quad B^y} \qquad (A \neq B)$$

The transformation is applied throughout a proof, through abstractions and applications (which are split, by Currying, into multiple), and is *linear* and so *terminating*. The result is a derivation for a CC-term where every variable has a uniform type. A contraction cannot be pushed past a ($\geq$) instance of idempotence on the same type. This has the form below left.

We translate the remaining derivation into one for a simply-typed non-deterministic λ-calculus, with a regular contraction rule and a type operator $\oplus$ with a co-diagonal rule from $A \oplus A$ to A. The result is below right.

$$\dfrac{\dfrac{\Gamma_1 \quad \cdots \quad \Gamma_n}{A \quad\quad A}}{\dfrac{A+A}{A \wedge A}\,\triangle}\,\geq \quad \mapsto \quad \dfrac{\dfrac{\Gamma_1 \quad \cdots \quad \Gamma_n}{A \quad\quad A}}{\dfrac{A}{A \wedge A}\,\triangle}\,\oplus$$

This derivation is for a typed, non-deterministic λ-term, which is strongly normalizing, and simulates reduction in both CC-terms. Consequently, these are strongly normalizing. ◀

E Omitted proofs and lemmas in Section 6

▶ **Theorem 22.** *A structural λ-term t is weakly [strongly] normalizing if and only if there is a typed, [strong,] uniform CC-term $u \succ t$.*

Proof sketch. $\Rightarrow$ Standard: normal forms can be typed, and subject expansion holds.
$\Leftarrow$ By Theorem 17 the typed CC-term s is strongly normalizing; hence t is strongly normalizing if weakened terms are typed, and weakly normalizing if weakened terms remain untyped. ◀