Electronic Notes in Theoretical Informatics And Computer Science

ENTICS
https://entics.episciences.org

Volume 5
Proceedings of MFPS 2025

Strong Normalization Through Idempotent Intersection Types: A New Syntactical Approach[⋆,⋆⋆]

Pablo Barenbaum[c,b,1] Simona Ronchi Della Rocca[a,2] Cristian Sottile[b,c,3]

[a] *Dipartimento di Informatica, Università di Torino, Italy*
[b] *ICC, CONICET—Universidad de Buenos Aires, Argentina*
[c] *Universidad Nacional de Quilmes, Argentina*

Abstract

It is well-known that intersection type assignment systems can be used to characterize strong normalization (SN). Typical proofs that typable lambda-terms are SN in these systems rely on *semantical* techniques. In this work, we study $\Lambda^{\mathtt{e}}_{\cap}$, a variant of Coppo and Dezani's (Curry-style) intersection type system and we propose a syntactical proof of strong normalization for it. We first design $\Lambda^{\mathtt{i}}_{\cap}$, a Church-style version of $\Lambda^{\mathtt{e}}_{\cap}$, in which terms closely correspond to typing derivations. We then prove that typability in $\Lambda^{\mathtt{i}}_{\cap}$ implies SN through a measure that, given a term, produces a natural number that decreases along with reduction. This measure provides a *syntactical* proof of SN. Finally, the result is extended to $\Lambda^{\mathtt{e}}_{\cap}$, since the two systems simulate each other.

Keywords: Lambda calculus, Intersection types, Strong normalization

1 Introduction

Intersection types were introduced by Coppo and Dezani [12], with the explicit aim of increasing the typability power of simple types. Beyond the arrow connective of simple types ($\to$), these systems incorporate an intersection connective ($\wedge$), which is commutative, associative and idempotent (*i.e.* $A \wedge A \equiv A$), which means that intersection is essentially a notation for *sets* of types.

Intersection type assignment systems come in different versions, and are able to characterize semantical properties of terms. For example, Coppo and Dezani's original system *characterizes* strong normalization, in the sense that a term is typable if and only if it is strongly normalizing. Recall that a λ-term is *strongly normalizing* if every reduction sequence starting from it eventually terminates.

There are many other intersection type systems characterizing other properties besides strong normalization, such as head normalization [13,24], where we recall that a λ-term is *head normalizing* if head

⋆ Funded by the European Union through the MSCA SE project QCOMICAL (Grant Agreement ID: 101182520).
⋆⋆We would like to thank the anonymous reviewers for their careful reading of the manuscript and their insightful comments, which helped improve the quality of this paper.
[1] Email: pbarenbaum@dc.uba.ar
[2] Email: ronchi@di.unito.it
[3] Email: csottile@dc.uba.ar

Rec'd Apr 3,2025; Pub'd Dec 15,2025 Proceedings Available Online at 10.46298/entics.16693 ENTICS MFPS 41 Proceedings

reduction (reducing at every step the redex in head position) eventually terminates. Furthermore, intersection types are a very powerful tool to reason about the interpretation of terms in various models of λ-calculus [3,30,5].

The existent proofs of the characterization properties are not straightforward: for both strong and head normalization, the direction "typable implies normalizing" is generally carried out using semantical tools like computability or reducibility candidates [29,24]. These semantical proofs are in some sense standard, but it is not easy to extract from them an intuition about what is going on. Having an explicit measure that decreases with β-reduction would be preferable. In the case of strong normalization, there are three completely syntactical proofs, based on a decreasing measure [22,9,11]; we will briefly discuss them and their relation with the present work at the end of this section.

A non-idempotent version of intersection types has been defined [20,21,15], where the intersection connective is *not* idempotent (*i.e.* $A \wedge A \not\equiv A$), and thus becomes a notation for *multisets* of types. The non-idempotent systems preserve some of the properties of the idempotent ones: they can characterize strong and head normalization [7]. From a semantical point of view, they can describe relational models of λ-calculus [28]; even if this class of models is quite poor, with respect to the λ-theories that they induce, they are useful to explore quantitative properties of typable terms. Unlike what happens in the idempotent case, the proof that these systems with non-idempotent intersection characterize strong and head normalization can be carried out by an easy induction on the size of the typing derivation, which decreases at every reduction step. This relies on the quantitative nature of non-idempotency, and in particular on the fact that, in a derivation, the cardinality of the multiset of types of a variable x is an upper bound of the number of occurrences of x in the typed term. Given that these proofs rely crucially on the fact that intersection is non-idempotent, there is not much hope of extending them to the idempotent case. But we think that it would be interesting to explore if there is a decreasing measure for idempotent intersection types which is simpler than those in [22,9]. Here we present a positive answer to this problem, namely a decreasing measure consisting of just a natural number.

1.1 Contributions

Recently, a proof of strong normalization for simply typed λ-calculus has been presented [2], based on a decreasing measure which can be defined constructively, by relying on a non-erasing notion of reduction inspired by *memory calculi* [26,23,16,27,10]. Namely, each β-reduction step creates a "wrapper" containing a copy of the argument, remembering all the erased subterms. An operation called *full simplification* is defined to repeatedly perform the complete development of all the redexes of maximum degree in a term [4]. A measure decreasing at every reduction step is then given simply by a natural number corresponding to the number of wrappers that remain after full simplification. Here we **adapt this technique to idempotent intersection types**.

The first step in order to do this job is to design an intrinsically typed version of intersection types, which are usually presented as extrinsically typed systems instead. Let us recall that, in an *extrinsically typed* (Curry-style) system, terms are untyped and there is a set of typing rules assigning types to them, whereas in an *intrinsically typed* (Church-style) system, terms are decorated with types, and typing rules are just a check of syntactical correctness. There are some attempts in the literature in this direction, *e.g.* in [8,25,1], but they are not suitable for the application of the discussed techniques. In fact the big difference between simple and intersection types, on which the power of the latter relies, is that, if M is a term of λ-calculus typable by intersection types and N is a subterm of M occurring once in it, a typing derivation of a judgement $\Gamma \vdash_{\mathsf{e}} M : A$ can contain $n \geq 1$ subderivations with object N. Since N can contain redexes, performing a reduction inside N corresponds to performing n reductions *in parallel* on the derivation. The natural way to achieve this is to equip the Church-style intersection type system with a parallel notion of reduction, and this is indeed what all the existing proposals do. At the same time, in order to apply the cited method, we would need to reduce these subderivations independently from each other.

[4] In this context, the *degree* of a redex $(\lambda x.\, t)\, s$ is the height of the type of the abstraction $\lambda x.\, t$.

Our starting point is an **e**xtrinsically typed subset of Λ, called $\Lambda^{\mathtt{e}}_{\cap}$, defined through an intersection type assignment system. This system is a variant of well-known systems in the literature that characterize strong normalization. In order to build an intrinsically typed version of $\Lambda^{\mathtt{e}}_{\cap}$, we exploit the idea that idempotent intersection represents a set of types, and we explicitly use sets both in the grammar of types and terms of the typed language, so for example if t and s are terms of type A and B respectively, then $\{t, s\}$ is a set of terms which can be assigned the set of types $\{A, B\}$. Following this idea, we formulate an **i**ntrinsically typed version of $\Lambda^{\mathtt{e}}_{\cap}$ called $\Lambda^{\mathtt{i}}_{\cap}$, where sets of types are decorations for sets of terms. While these two systems can be dealt with independently from each other, they are shown to *simulate* each other. The new typed language $\Lambda^{\mathtt{i}}_{\cap}$ enjoys all the good properties we expect, such as subject reduction and confluence. More importantly, it is suitable for the application of a method extending that of [2]. As a matter of fact, using the notion of complete development by degree, and relying on an auxiliary *memory calculus* $\Lambda^{\mathtt{im}}_{\cap}$, we define a measure which is a natural number decreasing at every reduction step in $\Lambda^{\mathtt{i}}_{\cap}$. Then, we prove that, for every term M in $\Lambda^{\mathtt{e}}_{\cap}$, there is a term t in $\Lambda^{\mathtt{i}}_{\cap}$ such that an infinite reduction sequence in M entails an infinite reduction sequence in t, thus obtaining a new proof of strong normalization for $\Lambda^{\mathtt{e}}_{\cap}$. Finally, we complete the picture by proving that $\Lambda^{\mathtt{i}}_{\cap}$ gives type to all the strongly normalizing terms, hence showing that it has the same typability power of the original paper [12].

1.2 Comparison with related works

As we recalled before, there are three other syntactical proofs of strong normalization for idempotent intersection types, namely [22,9,11]. Both [22,9] define an intermediate language, typed by intersection types, equipped with a suitable non-erasing reduction rule. So, as known for λI (*i.e.* non–erasing) based systems [4, §11.3], weak normalization there entails strong normalization. Using the property that the normalization of both the reductions implies β-normalization, they prove weak normalization for terms typable in the intersection type assignment system, and obtain strong normalization as a corollary. In both papers, the normalization proofs are carried out by defining a measure that decreases when a particular reduction strategy is chosen. The approach of reducing weak to strong normalization can be traced back to at least Nederpelt's and Klop's theses [26,23].

In [22] the intermediate language is an intrinsically typed λ-calculus extended with lists, the reduction rule is the composition of a parallel β-rule and a commutation rule, and the measure is a multiset of natural numbers decreasing under the innermost reduction. In [9] the intermediate language is a variant of the Klop's calculus, extrinsecally intersection typed, and the reduction is the β-reduction, but copying its argument when it is going to be erased; the weak normalization proof is obtained through the weak cut-elimination property of the type derivation, and the measure is a pair of natural numbers, decreasing under the strategy choosing the innermost redex of maximum degree.

Since our work uses an approach in some sense similar to [22,9], we first highlight the novelties that we consider important with respect to them.

- We share with [22] the use of an intermediate language which is intrinsically typed by intersection types, but while their language is *ad hoc*, and its properties are not explored, $\Lambda^{\mathtt{i}}_{\cap}$ is shown here to be the Church version of the Curry-style system. Indeed, $\Lambda^{\mathtt{e}}_{\cap}$ and $\Lambda^{\mathtt{i}}_{\cap}$ simulate each other, and reduction corresponds to cut-elimination in the derivation tree; we consider this to be an important result on its own.
- We do not need to define an *ad hoc* reduction rule; $\Lambda^{\mathtt{i}}_{\cap}$ uses directly the β-reduction, in its typed version.
- The codomain of our measure is simpler than multisets and pairs in [22,9], consisting of a single natural number.
- Our technique provides a measure that decreases for *every* reduction strategy. There are proofs with these characteristics for the λ-calculus, such as Gandy's proof and its derivatives [19,17], but not for intersection types, as far as we know.

Despite these differences, their works and ours rely fundamentally on Turing's remark that contracting a redex cannot create redexes of higher or equal degree [18,6], which is the basis for Turing's proof of weak normalization of the simply typed λ-calculus [18].

The approach of [11] is completely different: indeed, the normalization of intersection types here is not

the main result, but it is an intermediate step in the proof that computational power of intersection types is the same as that of simple types. The authors define an embedding from intersection type to simple type derivations which commutes with the β-reduction. So they do not define any decreasing measure. Let us notice that the embedding translates terms to linear terms, and that the intersection is used in a non commutative way, since the translation of $A \wedge B$ does not coincide with that of $B \wedge A$.

One could define a measure for intersection types by combining the translation with one of the existing measures for simply typed λ-calculus, for instance the measure $\mathcal{W}$ in [2]. Due to information lost in the translation, this would supply a natural number greater or equal than the one obtained in this paper. Since decreasing measures serve as an upper bound for the longest reduction chain of terms, a tighter measure provides a more accurate computational analysis and would therefore be generally preferred.

It is worth noting that even the combination with a simply typed λ-calculus exact measure (*e.g.* [17]) would not yield a tighter (nor looser) measure than $\mathcal{W}$. The loss of intersection type information that occurs during the translation is a hard limitation for any measure based on it; they cannot be refined to exactness. Our more coupled approach, since it operates *within* the intersection system, provides a finer analysis of reduction and thus allows for further improvement, which is indeed one of the proposed future works. We consider the refinability of our measure an advantage over measures based on such translation.

Organization of this paper

Section 2 presents the languages $\Lambda^{\mathtt{e}}_{\cap}$ and $\Lambda^{\mathtt{i}}_{\cap}$, their relations and properties. Section 3 constructs a decreasing measure that entails strong normalization of $\Lambda^{\mathtt{i}}_{\cap}$ by means of an auxiliary memory calculus $\Lambda^{\mathtt{im}}_{\cap}$. Finally, Section 4 shows how this can be used to prove strong normalization of $\Lambda^{\mathtt{e}}_{\cap}$, concludes, and proposes future works.

2 An intrinsically typed presentation of idempotent intersection types

This section starts by presenting $\Lambda^{\mathtt{e}}_{\cap}$ (2.1), an intersection type assignment system for λ-terms, closely related to other well-known systems that characterize strong normalization. In Section 2.2, we formulate an intrinsically typed version of the system, called $\Lambda^{\mathtt{i}}_{\cap}$, and we study its properties, including subject reduction and confluence. In Section 2.3 we establish a formal proof-theoretical connection between $\Lambda^{\mathtt{e}}_{\cap}$ and $\Lambda^{\mathtt{i}}_{\cap}$, and we prove that $\Lambda^{\mathtt{e}}_{\cap}$ gives type to all the strongly normalizing terms. Recall that the set of untyped λ-terms Λ is defined as usual by the grammar $M ::= x \mid \lambda x.M \mid M\,M$. The sets of **types** $(A, B, \ldots)$ and **set-types** $(\vec{A}, \vec{B}, \ldots)$ are given mutually inductively as follows:

$$A ::= a \mid \vec{A} \to A \qquad (\vec{A} \neq \varnothing) \qquad\qquad \vec{A} ::= \{A_j\}_{j \in J}$$

where $a, b, \ldots$ range over a denumerable set of base types, and $\vec{A}$ stands for a finite set of types [5], where J denotes a finite set of indices, and $\{A_j\}_{j\in J}$ is the set $\{A_j \mid j \in J\}$. When we write a set-type, we keep the invariant that there are no repetitions, *i.e.* that if $h \neq k$ then $A_h \neq A_k$, so that the set is of cardinality $|J|$. Sometimes we will use the explicit notation $\{A_1, ..., A_n\}$. Note that a set-type is a finite set of types that may be empty, but the domain of an arrow type is always a *non-empty* set-type.

Sometimes we write $A \to B$ for $\{A\} \to B$. A typing context $(\Gamma, \Delta, \ldots)$ is a function mapping each variable to a set-type such that $\Gamma(x) \neq \varnothing$ for finitely many variables x. We write $\mathsf{dom}(\Gamma)$ for the set of variables which Γ assigns to a non-empty set-type. We adopt the standard notation for typing contexts, *e.g.* writing $x_1 : \vec{A}_1, \ldots, x_n : \vec{A}_n$ for the context Γ such that $\Gamma(x_i) = \vec{A}_i$ for each $i \in 1..n$ and $\Gamma(y) = \varnothing$ for every $y \notin \{x_1, \ldots, x_n\}$. We write $\Gamma \cup \Delta$ for the typing context such that $(\Gamma \cup \Delta)(x) = \Gamma(x) \cup \Delta(x)$ for every x, and we write $\Gamma \subseteq \Delta$ if $\Gamma(x) \subseteq \Delta(x)$ for every x. Note in particular that $\Gamma \subseteq (\Gamma, x : \vec{A})$ if $x \notin \mathsf{dom}(\Gamma)$. If $\mathcal{J}$ is a judgement, we write $\Pi \rhd \mathcal{J}$ to mean that Π is a particular derivation proving $\mathcal{J}$.

[5] Note that $\vec{A} ::= \{A_j\}_{j\in J}$ is not intended to be a grammatical production that determines a concrete syntax for finite sets, but rather as a formation rule meaning that a set-type $\vec{A}$ is given by a finite set of types.

2.1 Extrinsically typed system

We define a Curry-style intersection type assignment system, which is a variant of well-known systems in the literature, starting with the one introduced by Coppo and Dezani [12].

Definition 2.1 (The $\Lambda^{\mathtt{e}}_{\cap}$ type assignment system) *The $\Lambda^{\mathtt{e}}_{\cap}$ calculus is defined by means of two forms of judgement:*

1. $\Gamma \vdash_{\mathtt{e}} M : A$ *meaning that the λ-term M has type A under the context Γ.*
2. $\Gamma \Vdash_{\mathtt{e}} M : \vec{A}$ *meaning that the λ-term M has set-type $\vec{A}$ under the context Γ.*

The typing rules are as follows:

$$\frac{B \in \vec{A}}{\Gamma, x : \vec{A} \vdash_{\mathtt{e}} x : B}\ \mathtt{e\text{-}var} \qquad \frac{(\Gamma \vdash_{\mathtt{e}} N : A_j)_{j\in J} \quad (\forall h,k \in J.\ h \neq k \implies A_h \neq A_k)}{\Gamma \Vdash_{\mathtt{e}} N : \{A_j\}_{j\in J}}\ \mathtt{e\text{-}many}$$

$$\frac{\Gamma, x : \vec{A} \vdash_{\mathtt{e}} M : B}{\Gamma \vdash_{\mathtt{e}} \lambda x.\, M : \vec{A} \to B}\ \mathtt{e\text{-}I}{\to} \qquad \frac{\Gamma \vdash_{\mathtt{e}} M : \vec{A} \to B \quad \vec{A} \neq \varnothing \quad \Gamma \Vdash_{\mathtt{e}} N : \vec{A}}{\Gamma \vdash_{\mathtt{e}} M\, N : B}\ \mathtt{e\text{-}E}{\to}$$

Note that we restrict the codomain of contexts so that $\Gamma(x) \neq \varnothing$, so rule e-I$\to$ *cannot introduce an empty set to the left of the arrow.*

Example 2.2 *Let $\Gamma = x : \{\{A,B\} \to C, A, B\}$. Then:*

$$\dfrac{\dfrac{\{A,B\} \to C \in \Gamma(x)}{\Gamma \vdash_{\mathtt{e}} x : \{A,B\} \to C}\ \mathtt{e\text{-}var} \quad \dfrac{\dfrac{A \in \Gamma(x)}{\Gamma \vdash_{\mathtt{e}} x : A}\ \mathtt{e\text{-}var} \quad \dfrac{B \in \Gamma(x)}{\Gamma \vdash_{\mathtt{e}} x : B}\ \mathtt{e\text{-}var}}{\Gamma \Vdash_{\mathtt{e}} x : \{A,B\}}\ \mathtt{e\text{-}many}}{\dfrac{\Gamma \vdash_{\mathtt{e}} x\, x : C}{\varnothing \vdash_{\mathtt{e}} \lambda x.\, x\, x : \{\{A,B\} \to C, A, B\} \to C}\ \mathtt{e\text{-}I}{\to}}\ \mathtt{e\text{-}E}{\to}$$

Remark 2.3 *The system $\Lambda^{\mathtt{e}}_{\cap}$ presented above is not exactly the same system as that of [12], for two reasons. First, $\Lambda^{\mathtt{e}}_{\cap}$ uses sets instead of intersections, writing $\{A,B\} \to C$ rather than $(A \wedge B) \to C$. This can be understood only as a difference in notation, given that the intersection type constructor of [12] is assumed to be associative, commutative, and idempotent. Second, $\Lambda^{\mathtt{e}}_{\cap}$ allows sets (*i.e. *intersections) to appear only to the* left *of the arrow ($\to$) connective. For example, $(A \wedge B) \to (C \wedge D)$ is a well-formed type in the system of [12], whereas $\{A,B\} \to \{C,D\}$ is* ***not*** *a well-formed type in $\Lambda^{\mathtt{e}}_{\cap}$. This means that types in $\Lambda^{\mathtt{e}}_{\cap}$ are* strict types*, in the sense introduced by van Bakel [31]. Despite these two differences, we will prove that the original system of [12] and $\Lambda^{\mathtt{e}}_{\cap}$ are equivalent in typability power, as they both characterize strongly normalizable λ-terms.*

2.2 Intrinsically typed system

We define a Church-style presentation of the system, inspired by the linearization proposed by Kfoury [21].

Definition 2.4 (The $\Lambda^{\mathtt{i}}_{\cap}$ type assignment system) *The sets of* **terms** *$(t, s, \ldots)$ and* **set-terms** *$(\vec{t}, \vec{s}, \ldots)$ are given by the following grammar:*

$$t ::= x^A \mid \lambda x^{\vec{A}}.\, t \mid t\, \vec{t} \qquad\qquad \vec{t} ::= \{t_j\}_{j\in J}$$

where J stands for a finite set of indices and $\{t_j\}_{j\in J}$ is the set $\{t_j \mid j \in J\}$. Note that set-terms are finite sets of terms. When we write a set-term, we keep the invariant that there are no repetitions, i.e. *that if $h \neq k$ then $t_h \neq t_k$. The $\Lambda^{\mathtt{i}}_{\cap}$ calculus is defined by means of two forms of judgement:*

1. $\Gamma \vdash_{\mathtt{i}} t : A$ *meaning that a term t is of type A under the context Γ.*

2. $\Gamma \Vdash_{\mathtt{i}} \vec{t} : \vec{A}$ *meaning that a set-term* $\vec{t}$ *is of set-type* $\vec{A}$ *under the context* Γ.

The typing rules are as follows:

$$\frac{B \in \vec{A}}{\Gamma, x : \vec{A} \vdash_{\mathtt{i}} x^B : B}\ \mathtt{i\text{-}var} \qquad \frac{(\Gamma \vdash_{\mathtt{i}} t_j : A_j)_{j\in J} \quad (\forall h,k \in J.\, h \neq k \implies A_h \neq A_k)}{\Gamma \Vdash_{\mathtt{i}} \{t_j\}_{j\in J} : \{A_j\}_{j\in J}}\ \mathtt{i\text{-}many}$$

$$\frac{\Gamma, x : \vec{A} \vdash_{\mathtt{i}} t : B}{\Gamma \vdash_{\mathtt{i}} \lambda x^{\vec{A}}.\, t : \vec{A} \to B}\ \mathtt{i\text{-}I}{\to} \qquad \frac{\Gamma \vdash_{\mathtt{i}} t : \vec{A} \to B \qquad \Gamma \Vdash_{\mathtt{i}} \vec{s} : \vec{A} \qquad \vec{A} \neq \varnothing}{\Gamma \vdash_{\mathtt{i}} t\, \vec{s} : B}\ \mathtt{i\text{-}E}{\to}$$

A term t *is* **typable** *if* $\Gamma \vdash_{\mathtt{i}} t : A$ *holds for some* Γ, A. *Similarly, a set-term is typable if* $\Gamma \Vdash_{\mathtt{i}} \vec{t} : \vec{A}$ *holds for some* $\Gamma, \vec{A}$. *Unless otherwise specified, when we speak of* term *(resp.* set-term*) we mean* typable term *(resp.* typable set-term*). We write* $\Lambda^{\mathtt{i}}_{\cap}$ *for the set of typable terms,* i.e. $\Lambda^{\mathtt{i}}_{\cap} := \{t \mid t \text{ is typable}\}$. *The notions of free and bound occurrences of variables are defined as usual. The set of* **free variables of type** A *of a term* t *is written* $\mathsf{fv}_A(t)$ *and defined as the set of variables* x *such that* x^A *occurs free in* t. *For example,* $\mathsf{fv}_A(\lambda y^{\{\{A,B\}\to C\}}.\, y^{\{A,B\}\to C}\{x^A, z^B\}) = \{x\}$ *when* $A \neq B$. *The set of* **free variables** *of a term* t *is written* $\mathsf{fv}(t)$. *Terms are considered up to* α*-renaming of bound variables.*

Notation 2.5 *Sometimes we write* $t\, s$ *to stand for* $t\,\{s\}$. *We may omit type annotations over variables if they are clear from the context.*

Remark 2.6 *Note that, in the rule* `i-many`*, we explicitly require that the function* $j \mapsto A_j$ *is injective (*i.e. $\forall h,k \in J.\, h \neq k \implies A_h \neq A_k$*). For example, let* $\Gamma = (x : \{A, B\}, y : \{A\})$. *Then the following are* ***valid*** *judgments:*

$$\Gamma \Vdash_{\mathtt{i}} \{x^A\} : \{A\} \qquad \Gamma \Vdash_{\mathtt{i}} \{x^A, x^B\} : \{A, B\} \qquad \Gamma \Vdash_{\mathtt{i}} \{x^B, y^A\} : \{A, B\}$$

On the other hand, the following judgments are ***not valid****:*

$$\Gamma \Vdash_{\mathtt{i}} \{x^A, y^A\} : \{A\} \qquad \Gamma \Vdash_{\mathtt{i}} \{x^A, x^A\} : \{A\}$$

Example 2.7 *The derivation of Example 2.2 can be encoded in* $\Lambda^{\mathtt{i}}_{\cap}$ *as the following term, which has type* $\{\{A, B\} \to C, A, B\} \to C$ *in the empty context:*

$$\lambda x^{\{\{A,B\}\to C, A, B\}}.\, x^{\{A,B\}\to C}\,\{x^A, x^B\}$$

Remark 2.8 *Note that, in general, a set-term may be an* empty *set of terms, but the argument of an application is always a* non-empty *set-term. We let set-types and set-terms to be empty to be able to write expressions such as* $\vec{t} = \{t\} \cup \vec{s}$ *to mean that* $\vec{t}$ *is a set containing at least one element* t*, leaving the possibility that* $\vec{s}$ *may be empty.*

Remark 2.9 (Subterm property) *If* t *is typable, all of its subterms are typable.*

Lemma 2.10 (Weakening and strengthening)

1. *If* $\Gamma \vdash_{\mathtt{i}} t : A$ *(resp.* $\Gamma \Vdash_{\mathtt{i}} \vec{t} : \vec{A}$*) and* $\Gamma \subseteq \Delta$ *then* $\Delta \vdash_{\mathtt{i}} t : A$ *(resp.* $\Delta \Vdash_{\mathtt{i}} \vec{t} : \vec{A}$*).*
2. *If* $\Gamma, x : \{A\} \cup \vec{B} \vdash_{\mathtt{i}} t : C$ *(resp.* $\Gamma, x : \{A\} \cup \vec{B} \Vdash_{\mathtt{i}} \vec{t} : \vec{C}$*) and* $x \notin \mathsf{fv}_A(t)$*, then* $\Gamma' \vdash_{\mathtt{i}} t : C$ *(resp.* $\Gamma' \Vdash_{\mathtt{i}} \vec{t} : \vec{C}$*), where* Γ' *is* $(\Gamma, x : \vec{B})$ *if* $\vec{B} \neq \varnothing$*, and* Γ *otherwise.*

Definition 2.11 (Minimal typing context) *For each term* t *(resp. set-term* $\vec{t}$*), its* minimal typing context *is written* $\gamma(t)$ *(resp.* $\gamma(\vec{t})$*) and defined as follows:*

$$\begin{array}{llll} \gamma(x^A) & := x : \{A\} & \gamma(\lambda x^{\vec{A}}.\, t) & := \gamma(t) \setminus x \\ \gamma(t\,\vec{s}) & := \gamma(t) \cup \gamma(\vec{s}) & \gamma(\{t_1, \ldots, t_n\}) & := \cup_{i=1}^{n} \gamma(t_i) \end{array}$$

where $\Gamma \setminus x$ *is defined in such a way that* $(\Gamma \setminus x)(x) = \varnothing$ *and* $(\Gamma \setminus x)(y) = \Gamma(y)$ *for* $x \neq y$.

Lemma 2.12 (Type uniqueness) *If* $\Gamma_1 \vdash_{\mathtt{i}} t : A_1$ *and* $\Gamma_2 \vdash_{\mathtt{i}} t : A_2$ *then* $A_1 = A_2$. *Furthermore,* $\gamma(t) \subseteq \Gamma_1$ *and* $\gamma(t) \subseteq \Gamma_2$ *and* $\gamma(t) \vdash_{\mathtt{i}} t : A_1$.

Remark 2.13 *Sometimes we write* t^A *to mean that* $\gamma(t) \vdash_{\mathtt{i}} t : A$. *Note that, by the previous lemma, there is a unique* A *such that* t^A.

Observation 2.14 (Bijection between a set-term and its set-type) *Let* $\Gamma \Vdash_{\mathtt{i}} \vec{s} : \vec{A}$. *Given* $A' \in \vec{A}$, *we know by rule* `i-many` *that there exists a derivation* $\Gamma \vdash_{\mathtt{i}} s' : A'$. *This is the only derivation with type* A' *in the premises of the rule: otherwise, we would contradict the rule's side condition by having different derivations with the same type. Therefore,* s' *is unique. Given* $s' \in \vec{s}$, *we know by rule* `i-many` *that it has a type* A'; *by Lemma 2.12, this type is unique.*

Definition 2.15 (Substitution in $\Lambda^{\mathtt{i}}_{\cap}$) *We define operations of capture-avoiding substitution for terms and set-terms by mutual recursion. Let* $\Gamma \Vdash_{\mathtt{i}} \vec{s} : \vec{A}$ *with* $\vec{s} = \{s_i\}_{i \in I}$, $\vec{A} = \{A_i\}_{i \in I}$, *and* $\Gamma \vdash_{\mathtt{i}} s_i : A_i$ *for all* $i \in I$. *we define* $t[x^{\vec{A}} := \vec{s}]$ *recursively as follows.*

$$
\begin{aligned}
x^{A_i}[x^{\vec{A}} := \vec{s}] &:= s_i \\
y^B[x^{\vec{A}} := \vec{s}] &:= y^B \\
(\lambda y^{\vec{B}}.\, t)[x^{\vec{A}} := \vec{s}] &:= \lambda y^{\vec{B}}.\, t[x^{\vec{A}} := \vec{s}] \\
(t\ \vec{u})[x^{\vec{A}} := \vec{s}] &:= t[x^{\vec{A}} := \vec{s}]\ \vec{u}[x^{\vec{A}} := \vec{s}] \\
\{t_1, \ldots, t_n\}[x^{\vec{A}} := \vec{s}] &:= \{t_1[x^{\vec{A}} := \vec{s}], \ldots, t_n[x^{\vec{A}} := \vec{s}]\}
\end{aligned}
$$

Note that, in the first case, s_i *exists and is unique due to Observation 2.14.*

Lemma 2.16 (Substitution Lemma) *Let* $\Gamma, x : \vec{A} \vdash_{\mathtt{i}} t : B$ *and* $\Gamma \Vdash_{\mathtt{i}} \vec{s} : \vec{A}$. *Then* $\Gamma \vdash_{\mathtt{i}} t[x^{\vec{A}} := \vec{s}] : B$.

Proof. By induction on t, generalizing the statement to set-terms, where the interesting cases are:

1. If $t = x^B$, then $\Gamma, x : \vec{A} \vdash_{\mathtt{i}} x^B : B$ with $B \in \vec{A}$. By Observation 2.14, there exists a unique $s_i \in \vec{s}$ such that $\Gamma \vdash_{\mathtt{i}} s_i : B$. Therefore, $x^B[x^{\vec{A}} := \vec{s}] = s_i$ of type B.
2. If $\vec{t} = \{t_1, \ldots, t_n\}$, then B is indeed a set-type $\vec{B} = \{B_1, \ldots, B_n\}$, and $\Gamma, x : \vec{A} \Vdash_{\mathtt{i}} \{t_1, \ldots, t_n\} : \{B_1, \ldots, B_n\}$. By IH, $\Gamma \vdash_{\mathtt{i}} t_i[x^{\vec{A}} := \vec{s}] : B_i$ for all i such that $1 \leq i \leq n$, so by `i-many` we have $\Gamma \Vdash_{\mathtt{i}} \{t_1[x^{\vec{A}} := \vec{s}], \ldots, t_n[x^{\vec{A}} := \vec{s}]\} : \{B_1, \ldots, B_n\}$. Note that substitution preserves cardinality in typed set-term derivations. If we had $t_i[x^{\vec{A}} := \vec{s}] = t_j[x^{\vec{A}} := \vec{s}]$ with distinct i, j such that $1 \leq i, j \leq n$, they would have the same type by Lemma 2.12. This would imply that t_i and t_j have the same type, which is impossible by rule `i-many`.

□

Definition 2.17 (Reduction in $\Lambda^{\mathtt{i}}_{\cap}$) *A notion of reduction called* `i`*-reduction is defined over terms and set-terms by the following rule, closed by congruence under arbitrary contexts:*

$$(\lambda x^{\vec{A}}.\, t)\ \vec{s} \to_{\mathtt{i}} t[x^{\vec{A}} := \vec{s}]$$

Note in particular that congruence rules allow reducing inside the argument of an application, i.e. $\vec{s} \to_{\mathtt{i}} \vec{u}$ *implies* $t\ \vec{s} \to_{\mathtt{i}} t\ \vec{u}$, *and inside any of the elements of a set-term,* i.e. $t_i \to_{\mathtt{i}} t_i'$ *implies* $\{t_1, \ldots, t_i, \ldots, t_n\} \to_{\mathtt{i}} \{t_1, \ldots, t_i', \ldots, t_n\}$. *We write* $\to_{\mathtt{i}}^*$ *for the reflexive–transitive closure of* $\to_{\mathtt{i}}$.

Reduction in $\Lambda^{\mathtt{i}}_{\cap}$ enjoys the following properties. See Appendices A.1.1 and A.1.2 for detailed proofs.

Proposition 2.18 (Subject reduction) *If* $t \to_{\mathtt{i}} t'$ *and* $\Gamma \vdash_{\mathtt{i}} t : A$ *then* $\Gamma \vdash_{\mathtt{i}} t' : A$.

Proposition 2.19 (Confluence) *If* $t \to_{\mathtt{i}}^* s$ *and* $t \to_{\mathtt{i}}^* u$ *then there is* r *such that* $s \to_{\mathtt{i}}^* r$ *and* $u \to_{\mathtt{i}}^* r$.

2.3 Correspondence between Curry and Church-style systems

A typable term in the Church-style system $\Lambda^{\mathtt{i}}_{\cap}$ does not necessarily represent a derivation tree in the Curry-style system $\Lambda^{\mathtt{e}}_{\cap}$. To relate these systems, we consider the following notion:

Definition 2.20 (Refinement) *We define a binary relation $t \sqsubset M$ between typable terms $t \in \Lambda^{\mathtt{i}}_{\cap}$ and untyped λ-terms $M \in \Lambda$, as well as a binary relation $\vec{t} \sqsubset M$ between typable set-terms $\vec{t} \subset \Lambda^{\mathtt{i}}_{\cap}$ and untyped λ-terms $M \in \Lambda$, according to the inductive rules below. In that case, we say that t is a* **refinement** *of M.*

$$\frac{}{x^A \sqsubset x} \qquad \frac{t \sqsubset M}{\lambda x^A.\, t \sqsubset \lambda x.\, M} \qquad \frac{t \sqsubset M \quad \vec{s} \sqsubset N}{t\,\vec{s} \sqsubset M\,N} \qquad \frac{(t_i \sqsubset M)_{i\in 1..n} \quad n>0}{\{t_1,\dots,t_n\} \sqsubset M}$$

Observation 2.21

1. *If $t \sqsubset M_1$ and $t \sqsubset M_2$ then $M_1 = M_2$.*
2. *If $t \sqsubset M$ and $\vec{s} \sqsubset N$ then $t[x^{\vec{A}} := \vec{s}] \sqsubset M[x := N]$.*

Definition 2.22 (Uniformity and type erasure) *A typable term $t \in \Lambda^{\mathtt{i}}_{\cap}$ is* **uniform** *if there exists $M \in \Lambda$ such that $t \sqsubset M$. Similarly, a typable set-term $\vec{t} \subseteq \Lambda^{\mathtt{i}}_{\cap}$ is uniform if there exists $M \in \Lambda$ such that $\vec{t} \sqsubset M$.*

If $t \in \Lambda^{\mathtt{i}}_{\cap}$ is a uniform term, its **type erasure** *$t^{\mathtt{e}} \in \Lambda$ is defined as the unique λ-term such that $t \sqsubset t^{\mathtt{e}}$. Similarly, if $\vec{t} \subset \Lambda^{\mathtt{i}}_{\cap}$ is a uniform set-term, $\vec{t}^{\,\mathtt{e}} \in \Lambda$ is defined as the unique λ-term such that $\vec{t} \sqsubset \vec{t}^{\,\mathtt{e}}$. This can be written as a recursive definition:*

$$(x^A)^{\mathtt{e}} := x \qquad (\lambda x^{\vec{A}}.\, t)^{\mathtt{e}} := \lambda x.\, t^{\mathtt{e}} \qquad (t\,\vec{s})^{\mathtt{e}} := t^{\mathtt{e}}\,\vec{s}^{\,\mathtt{e}} \qquad \{s_1,\dots,s_n\}^{\mathtt{e}} := s_1^{\mathtt{e}}$$

If $t \in \Lambda^{\mathtt{i}}_{\cap}$ is uniform, for each derivation $\Pi \rhd \Gamma \vdash_{\mathtt{i}} t : A$, we construct a derivation $\Pi^{\mathtt{e}} \rhd \Gamma \vdash_{\mathtt{e}} t^{\mathtt{e}} : A$ by erasing all the type annotations in terms. Similarly, each derivation $\Pi \rhd \Gamma \Vdash_{\mathtt{i}} \vec{t} : \vec{A}$ is mapped to a derivation $\Pi^{\mathtt{e}} \rhd \Gamma \Vdash_{\mathtt{e}} \vec{t}^{\,\mathtt{e}} : \vec{A}$.

Definition 2.23 (Type decoration) *For each derivation $\Pi \rhd \Gamma \vdash_{\mathtt{e}} M : A$, we construct a term $t \in \Lambda^{\mathtt{i}}_{\cap}$ and a derivation $\Pi^{\mathtt{i}} \rhd \Gamma \vdash_{\mathtt{i}} t : A$ inductively as below. Similarly, for each derivation $\Pi \rhd \Gamma \Vdash_{\mathtt{e}} M : \vec{A}$ we construct a set-term $\vec{t} \in \Lambda^{\mathtt{i}}_{\cap}$ and a derivation $\Pi^{\mathtt{i}} \rhd \Gamma \Vdash_{\mathtt{i}} \vec{t} : \vec{A}$.*

$$\frac{B \in \vec{A}}{\Gamma, x : \vec{A} \vdash_{\mathtt{e}} x : B}\ \mathtt{e\text{-}var} \qquad \rightsquigarrow \frac{B \in \vec{A}}{\Gamma, x : \vec{A} \vdash_{\mathtt{i}} x^B : B}\ \mathtt{i\text{-}var}$$

$$\frac{\Gamma, x : \vec{A} \vdash_{\mathtt{e}} M : B}{\Gamma \vdash_{\mathtt{e}} \lambda x.\, M : \vec{A} \to B}\ \mathtt{e\text{-}I{\to}} \qquad \rightsquigarrow \frac{\Gamma, x : \vec{A} \vdash_{\mathtt{i}} t : B}{\Gamma \vdash_{\mathtt{i}} \lambda x^{\vec{A}}.\, t : \vec{A} \to B}\ \mathtt{i\text{-}I{\to}}$$

$$\frac{\Gamma \vdash_{\mathtt{e}} M : \vec{A} \to B \quad \Gamma \Vdash_{\mathtt{e}} N : \vec{A}}{\Gamma \vdash_{\mathtt{e}} M\,N : B}\ \mathtt{e\text{-}E{\to}} \rightsquigarrow \frac{\Gamma \vdash_{\mathtt{i}} t : \vec{A} \to B \quad \Gamma \Vdash_{\mathtt{i}} \vec{s} : \vec{A}}{\Gamma \vdash_{\mathtt{i}} t\,\vec{s} : B}\ \mathtt{i\text{-}E{\to}}$$

$$\frac{(\Gamma \vdash_{\mathtt{e}} N : A_i)_{i\in 1..n}}{\Gamma \Vdash_{\mathtt{e}} N : \{A_1,\dots,A_n\}}\ \mathtt{e\text{-}many} \qquad \rightsquigarrow \frac{(\Gamma \vdash_{\mathtt{i}} s_i : A_i)_{i\in 1..n}}{\Gamma \Vdash_{\mathtt{i}} \{s_1,\dots,s_n\} : \{A_1,\dots,A_n\}}\ \mathtt{i\text{-}many}$$

Theorem 2.24 (Correspondence)

1. *If $\Pi \rhd \Gamma \vdash_{\mathtt{e}} M : A$, then there exists a uniform $t \in \Lambda^{\mathtt{i}}_{\cap}$ such that $\Pi^{\mathtt{i}} \rhd \Gamma \vdash_{\mathtt{i}} t : A$ and $t \sqsubset M$. Moreover, $(\Pi^{\mathtt{i}})^{\mathtt{e}} = \Pi$.*

2. *If* $\Pi \rhd \Gamma \vdash_{\mathtt{i}} t : A$ *and* t *is uniform then* $\Pi^{\mathtt{e}} \rhd \Gamma \vdash_{\mathtt{e}} t^{\mathtt{e}} : A$. *Moreover,* $(\Pi^{\mathtt{e}})^{\mathtt{i}} = \Pi$.

Proof.

1. We generalize the statement to set-terms, *i.e.* if $\Pi \rhd \Gamma \Vdash_{\mathtt{e}} M : \vec{A}$ then there exists a uniform $\vec{t} \in \Lambda^{\mathtt{i}}_{\cap}$ such that $\Pi^{\mathtt{i}} \rhd \Gamma \Vdash_{\mathtt{i}} \vec{t} : \vec{A}$ and $\vec{t} \sqsubset M$. We proceed by induction on the derivation Π. The only not immediate case is when $\Pi \rhd \Gamma \Vdash_{\mathtt{e}} M : \vec{A}$. Then this judgement has been obtained by rule `e-many` with premises $(\Gamma \vdash_{\mathtt{e}} M : A_i)_{i \in 1..n}$; by IH there are derivations proving $\Gamma \Vdash_{\mathtt{e}} t_i : A_i$, where $t_i \sqsubset M$. Then, by definition, $\{t_1, \dots, t_n\} \sqsubset M$, and the proof follows by rule `i-many`. The other cases come directly from the definition of type decoration.
2. We generalize the statement to set-terms, *i.e.* if $\Pi \rhd \Gamma \Vdash_{\mathtt{i}} \vec{t} : \vec{A}$ and $\vec{t}$ is uniform then $\Pi^{\mathtt{e}} \rhd \Gamma \Vdash_{\mathtt{e}} \vec{t}^{\mathtt{e}} : \vec{A}$. We proceed by induction on the derivation Π. The only not immediate case is when $\Pi \rhd \Gamma \Vdash_{\mathtt{i}} \vec{t} : \vec{A}$. Let $\vec{t} = \{t_1, \dots, t_n\}$, then $t \sqsubset M$ implies, by definition of uniformity, $t_i \sqsubset M$. The judgement has been obtained by rule `i-many`, with premises $\Gamma \vdash_{\mathtt{i}} t_i : A_i$, so by IH, $\Gamma \vdash_{\mathtt{e}} t_i^{\mathtt{e}} : A_i$ where $t_i \sqsubset t_i^{\mathtt{e}}$. By Observation 2.21.1, $t_i \sqsubset t_i^{\mathtt{e}}$ and $t_i \sqsubset M$ imply $t_i^{\mathtt{e}} = M$, so $\Gamma \vdash_{\mathtt{e}} t : A_i$ and the result follows by applying rule `e-many`. The other cases come directly from the definition of type erasure.

□

Corollary 2.25 *Given a term* $M \in \Lambda$, *the following are equivalent:*

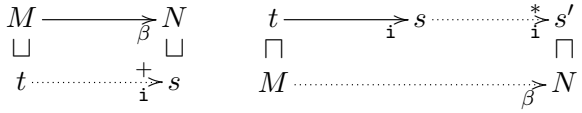
1. *There exist* Γ, A *such that* $\Gamma \vdash_{\mathtt{e}} M : A$ *holds.*
2. *There exists a term* $t \in \Lambda^{\mathtt{i}}_{\cap}$ *such that* $t \sqsubset M$.

Furthermore, $\to_\beta$ and $\to_{\mathtt{i}}$ simulate each other:

Theorem 2.26 (Simulation)

1. *If* $M \to_\beta N$ *and* $t \sqsubset M$, *there exists* $s \in \Lambda^{\mathtt{i}}_{\cap}$ *such that* $t \to^+_{\mathtt{i}} s$ *and* $s \sqsubset N$.
2. *If* $t \to_{\mathtt{i}} s$ *and* $t \sqsubset M$, *there exist* $N \in \Lambda$ *and* $s' \in \Lambda^{\mathtt{i}}_{\cap}$ *such that* $M \to_\beta N$ *and* $s \to^*_{\mathtt{i}} s'$ *and* $s' \sqsubset N$.

Graphically:



Proof.

1. We proceed by induction on M. The case $M = x$ cannot happen, in case $M = \lambda x.\, N$ the proof follows by IH. Let $M = P\,Q$, so $t = u\vec{r}$, where $u \sqsubset P$ and $\vec{r} \sqsubset Q$. The case $P \to_\beta P'$ is easy, by IH. Let $Q \to_\beta Q'$, and let $\vec{r} = \{r_1, ..., r_n\}$, where, by definition, $r_i \sqsubset Q$, for all $1 \leq i \leq n$. By IH, there are r'_i such that $r_i \to^+_{\mathtt{i}} r'_i$, such that $r'_i \sqsubset Q'$. So $\{r'_1, ..., r'_n\} \sqsubset Q'$ and $s = u\{r'_1, ..., r'_n\}$. Let $P = \lambda x.\, P'$, so $t = (\lambda x^A.\, t')\vec{r}$, where $t' \sqsubset P'$, and let $M \to_\beta P'[x := Q]$. By Observation 2.21.2, $t'[x^A := \vec{r}] \sqsubset P'[x := Q]$, and since $t \to^+_{\mathtt{i}} t'[x^A := \vec{r}]$, $s = t'[x^A := \vec{r}]$.
2. We generalize the statement to set-terms, *i.e.* if $\vec{t} \to_{\mathtt{i}} \vec{s}$ and $\vec{t} \sqsubset M$, there exist N and $\vec{s}'$ such that $M \to_\beta N$ and $\vec{s} \to^*_{\mathtt{i}} \vec{s}'$ and $\vec{s}' \sqsubset N$. The proof is by induction on terms. Let $\vec{t} = \{t_1, \dots, t_n, u\}$, and $\vec{s} = \{t_1, \dots, t_n, u'\}$, where $u \to_{\mathtt{i}} u'$. By IH, there are N and u'' such that $M \to_\beta N$, $u' \to^*_{\mathtt{i}} u''$ and $u'' \sqsubset N$. Note that $t_i \sqsubset M$ for all $i \in 1..n$. Now let $i \in 1..n$. By point 1, $M \to_\beta N$ and $t_i \sqsubset M$ imply there is t'_i such that $t_i \to^*_{\mathtt{i}} t'_i$ and $t'_i \sqsubset N$; then it suffices to take $\vec{s}' = \{t'_1, ..., t'_n, u''\}$. The other cases follow by IH, using Observation 2.21.2 in case $t = (\lambda x^A.\, t')\,\vec{u}$, and $s = t'[x^A := \vec{u}]$.

□

Some comments are in order. If N is a redex subterm of M, then a derivation tree for M in $\Lambda^{\mathtt{e}}_{\cap}$ may contain more than one subderivation with subject N. Contracting N with usual β-reduction corresponds to reducing in parallel *all* the occurrences of N in all the subderivations. In $\Lambda^{\mathtt{i}}_{\cap}$, this corresponds instead to performing the reductions one at a time.

Example 2.27 *Let $M = (\lambda x.\, x\, x)\,(I\, I)$, $A = B \to B$, and $I = \lambda x.\, x$. Consider the derivation Π:*

$$\dfrac{\dfrac{\dfrac{\vdots}{x : \{A \to A, A\} \vdash_{\mathtt{e}} x\, x : A}}{\vdash_{\mathtt{e}} \lambda x.\, x\, x : \{A \to A, A\} \to A}\ \mathtt{e\text{-}E}\!\to \quad \dfrac{\dfrac{\vdots}{\vdash_{\mathtt{e}} I\, I : A \to A} \quad \dfrac{\vdots}{\vdash_{\mathtt{e}} I\, I : A}}{\Vdash_{\mathtt{e}} I\, I : \{A \to A, A\}}\ \mathtt{e\text{-}many}}{\vdash_{\mathtt{e}} M : A}\ \mathtt{e\text{-}E}\!\to$$

Then we have that $\Pi^{\mathtt{i}} \rhd \vdash_{\mathtt{i}} t : A$, where $t = (\lambda x^{\{A\to A, A\}}.\, x\, x)\,\{(I\, I)^{A\to A}, (I\, I)^{A}\}$. Note that:

1. *t is a uniform term, and in particular $t \sqsubset M$.*
2. *$t \to_{\mathtt{i}} t_1 = (\lambda x^{\{A\to A, A\}}.\, x\, x)\,\{I^{A\to A}, (I\, I)^{A}\}$, where t_1 is not a uniform term.*
3. *$t \to_{\mathtt{i}} t_3 = (I\, I)^{A\to A}\,(I\, I)^{A}$ where $t_3 \sqsubset I\, I(I\, I)$ and $M \to_\beta I\, I(I\, I)$.*
4. *$t_1 \to_{\mathtt{i}}^{*} t_2 = (\lambda x^{\{A\to A, A\}}.\, x\, x)\,\{I^{A\to A}, I^{A}\}$, where t_2 is a uniform term; in particular $t_2 \sqsubset (\lambda x.\, x\, x)\, I$ and $M \to_\beta (\lambda x.\, x\, x) I$.*

Lemma 2.28 (Head subject expansion) *If $\Gamma \vdash_{\mathtt{i}} t[x^{\vec{A}} := \vec{s}]\vec{s}_1 ... \vec{s}_n : B$ and $\Gamma \Vdash_{\mathtt{i}} \vec{s} : \vec{A}$, then $\Gamma \vdash_{\mathtt{i}} (\lambda x^{\vec{A}}.\, t)\, \vec{s}\, \vec{s}_1 ... \vec{s}_n : B$.*

Proof. By induction on n. Let $n = 0$, so $\Gamma \vdash_{\mathtt{i}} t[x^{\vec{A}} := \vec{s}] : B$. We continue by induction on t. The interesting case is that of $t = y^B$, where $B \in \vec{A}$. Then $\Gamma, x^{\vec{A}} \vdash_{\mathtt{i}} x^B : B$, by rule (`i-var`), and $\Gamma \vdash_{\mathtt{i}} (\lambda x^{\vec{A}}.\, x^B)\, \vec{s} : B$ by rules (`i-I`$\to$) and (`i-E`$\to$). The remaining cases are straightforward, by IH and substitution definition. Let $n > 0$. Let $\Gamma \vdash_{\mathtt{i}} t[x^{\vec{A}} := \vec{s}]\vec{s}_1 ... \vec{s}_n\, \vec{u} : B$ and $\Gamma \Vdash_{\mathtt{i}} \vec{s} : \vec{A}$. Then, by the rules of the system, $\Gamma \vdash_{\mathtt{i}} t[x^{\vec{A}} := \vec{s}]\vec{s}_1 ... \vec{s}_n\, \vec{u} : B$ comes from $\Gamma \vdash_{\mathtt{i}} t[x^{\vec{A}} := \vec{s}]\vec{s}_1 ... \vec{s}_n : \vec{C} \to B$ and $\Gamma \Vdash_{\mathtt{i}} \vec{u} : \vec{C}$, for some $\vec{C}$. By IH, $\Gamma \vdash_{\mathtt{i}} (\lambda x^{A}.\, t)\, \vec{s}\, \vec{s}_1 ... \vec{s}_n : \vec{C} \to B$, and the result follows by rule (`i-E`$\to$). □

Lemma 2.29 (Strong Normalization typability) *Let $M \in \Lambda$ be strongly normalizing. Then there is $t \in \Lambda^{\mathtt{i}}_{\cap}$ such that $t \sqsubset M$ and $\Gamma \vdash_{\mathtt{i}} t : A$, for some Γ and A.*

Proof. The proof relies on the following inductive definition of the strongly normalizing terms (SN).

$$\dfrac{M_i \in SN \quad 1 \le i \le n}{x M_1 ... M_n \in SN}\ SN1 \qquad \dfrac{M \in SN}{\lambda x.\, M \in SN}\ SN2$$

$$\dfrac{M[x := N]\, M_1 ... M_n \in SN \quad N \in SN}{(\lambda x.\, M)\, N\, M_1 ... M_n \in SN}\ SN3$$

Let us consider rule $(SN1)$. By induction there are t_i such that $t_i \sqsubset M_i$ and $\Gamma_i \vdash_{\mathtt{i}} t_i : A_i$, so $\Gamma_i \Vdash_{\mathtt{i}} \vec{t_i} : \vec{A}_i$. So, by Lemma 2.10, and rule (`i-E`$\to$), $(\bigcup_{1\le i\le n} \Gamma_i) \cup x : \{\vec{A}_1 \to ... \to \vec{A}_n \to B\} \vdash_{\mathtt{i}} x^{\{\vec{A}_1 \to ... \to \vec{A}_n \to B\}} \vec{t_1} ... \vec{t_n} : B$. Let us consider rule $(SN3)$. Let $M[x := N]\, M_1 ... M_n$ and $N \in SN$. By IH, $\exists t, s, \Gamma, \Gamma', A, B$ such that $t \sqsubset M[x := N]\, M_1 ... M_n$, $s \sqsubset N$ and $\Gamma \vdash_{\mathtt{i}} t : B$, $\Gamma' \vdash_{\mathtt{i}} s : A$. Note that $\Gamma' \vdash_{\mathtt{i}} s : A$ implies $\Gamma' \Vdash_{\mathtt{i}} \vec{s} : \vec{A}$, where $\vec{s} = \{s\}$ and $\vec{A} = \{A\}$. $\Gamma \vdash_{\mathtt{i}} t : B$ and $t \sqsubset M[x := N]\, M_1 ... M_n$ imply $t = t'\, t_1 ... t_n$, where $t_i \sqsubset M_i$ and t' can be written as $t''[x^{\vec{A}} := \vec{s}]$. By Lemma 2.10, $\Gamma \cup \Gamma' \vdash_{\mathtt{i}} t''[x^{\vec{A}} := \vec{s}]\, t_1 ... t_n : B$ and $\Gamma \cup \Gamma' \Vdash_{\mathtt{i}} \vec{s} : \vec{A}$, and the result follows by Lemma 2.28. The case of rule $(SN2)$ follows directly by IH. □

3 Strong normalization via a decreasing measure

In this section, we prove strong normalization of the $\Lambda^{\mathtt{i}}_{\cap}$ calculus by providing an explicit decreasing measure, adapting the ideas behind the $\mathcal{W}$ measure of [2] to the setting of idempotent intersection type systems. The $\mathcal{W}$ measure is based on the fact that contracting a redex in the simply typed λ-calculus

cannot create a redex of higher or equal *degree*, where the degree of a redex is defined as the height of the type of its abstraction. This observation was already known to Turing, as reported by Gandy [19].

This section is organized as follows. In Section 3.1, we enrich the syntax of $\Lambda^{\mathtt{i}}_{\cap}$ by defining an auxiliary *memory* calculus $\Lambda^{\mathtt{im}}_{\cap}$ which incorporates *wrappers* $\langle\vec{s}\rangle$, and we study some technical properties that are needed later. In Section 3.2 we define an operation called *full simplification* for $\Lambda^{\mathtt{im}}_{\cap}$ terms, which iteratively contracts in parallel all redexes of maximum degree, showing that this yields the normal form of the term. Finally, in Section 3.3 we show that $\Lambda^{\mathtt{i}}_{\cap}$ is SN by defining a decreasing measure $\mathcal{W}$, which works by fully simplifying a term in $\Lambda^{\mathtt{im}}_{\cap}$ and counting the number of remaining wrappers.

3.1 The memory calculus $\Lambda^{\mathtt{im}}_{\cap}$

Definition 3.1 (The $\Lambda^{\mathtt{im}}_{\cap}$ calculus) *The sets of terms $(t, s, \ldots)$ and set-terms $(\vec{t}, \vec{s}, \ldots)$ are given by the following grammar:*

$$t ::= x^A \mid \lambda x^{\vec{A}}.\, t \mid t\,\vec{t} \mid t\langle\vec{t}\rangle \qquad\qquad \vec{t} ::= \{t_j\}_{j\in J}$$

where J stands for a finite set of indices. We write $\mathtt{L}$ for a list of wrappers, defined by the grammar $\mathtt{L} ::= \Box \mid \mathtt{L}\langle\vec{t}\rangle$, and $t\mathtt{L}$ for the term that results from extending t with all the wrappers in the list, i.e. *$t(\Box\langle\vec{s}_1\rangle\ldots\langle\vec{s}_n\rangle) = t\langle\vec{s}_1\rangle\ldots\langle\vec{s}_n\rangle$. Typing judgements are of the forms $\Gamma \vdash_{\mathtt{im}} t : A$ and $\Gamma \Vdash_{\mathtt{im}} \vec{t} : \vec{A}$, where types and typing contexts are defined as before. The typing rules extend the $\Lambda^{\mathtt{i}}_{\cap}$ type assignment system of Definition 2.4 with a typing rule* `im-wrap`*:*

$$\frac{B \in \vec{A}}{\Gamma, x : \vec{A} \vdash_{\mathtt{im}} x^B : B}\ \mathtt{im\text{-}var} \qquad \frac{\Gamma, x : \vec{A} \vdash_{\mathtt{im}} t : B}{\Gamma \vdash_{\mathtt{im}} \lambda x^{\vec{A}}.\, t : \vec{A} \to B}\ \mathtt{im\text{-}I}\to$$

$$\frac{\Gamma \vdash_{\mathtt{im}} t : \vec{A} \to B \qquad \Gamma \Vdash_{\mathtt{im}} \vec{s} : \vec{A} \qquad \vec{A} \neq \varnothing}{\Gamma \vdash_{\mathtt{im}} t\,\vec{s} : B}\ \mathtt{im\text{-}E}\to \qquad \frac{\Gamma \vdash_{\mathtt{im}} t : A \quad \Gamma \Vdash_{\mathtt{im}} \vec{s} : \vec{B}}{\Gamma \vdash_{\mathtt{im}} t\langle\vec{s}\rangle : A}\ \mathtt{im\text{-}wrap}$$

$$\frac{(\Gamma \vdash_{\mathtt{im}} t_j : A_j)_{j\in J} \quad (\forall h, k \in J.\ h \neq k \implies A_h \neq A_k)}{\Gamma \Vdash_{\mathtt{im}} \{t_j\}_{j\in J} : \{A_j\}_{j\in J}}\ \mathtt{im\text{-}many}$$

The operation of capture-avoiding substitution $t[x^{\vec{A}} := \vec{s}]$ is extended by declaring that $(t\langle\vec{u}\rangle)[x^{\vec{A}} := \vec{s}] = t[x^{\vec{A}} := \vec{s}]\langle\vec{u}[x^{\vec{A}} := \vec{s}]\rangle$. Reduction in the $\Lambda^{\mathtt{im}}_{\cap}$-calculus, called `im`*-reduction, is defined over terms and set-terms by the following rule, closed by congruence under arbitrary contexts:*

$$(\lambda x^{\vec{A}}.\, t)\mathtt{L}\,\vec{s} \to_{\mathtt{im}} t[x := \vec{s}]\langle\vec{s}\rangle\mathtt{L}$$

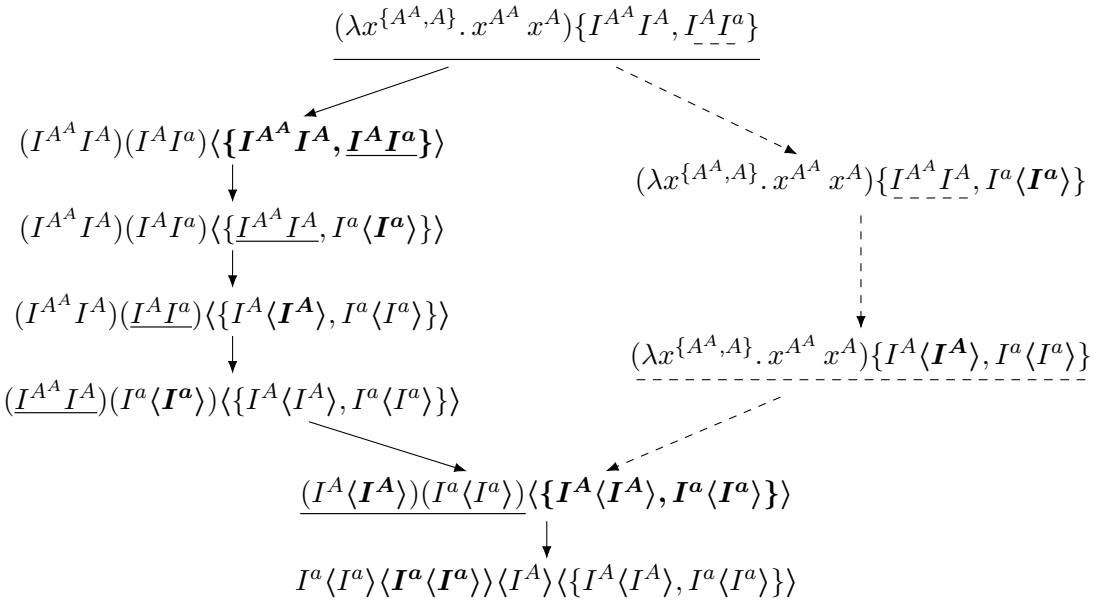
Abstractions followed by lists of wrappers, i.e. *terms of the form $(\lambda x.\, t)\mathtt{L}$ are called wrapped abstractions or* **w-abstractions** *for short. A* **redex** *is an expression matching the left-hand side of the* `im` *rule, which must be an applied w-abstraction. The* **height** *of a type A (resp. set-type $\vec{A}$) is written $\mathsf{h}(A)$ (resp. $\mathsf{h}(\vec{A})$) and defined as follows:*

$$\begin{aligned} \mathsf{h}(a) &:= 0 \\ \mathsf{h}(\vec{A} \to B) &:= 1 + \max\{\mathsf{h}(\vec{A}), \mathsf{h}(B)\} \\ \mathsf{h}(\{A_1, \ldots, A_n\}) &:= \max\{\mathsf{h}(A_1), \ldots, \mathsf{h}(A_n)\} \end{aligned}$$

We write $\mathsf{type}(t)$ for the type of t, which is uniquely defined for typable terms. The **degree** *of a w-abstraction is the height of its type. The degree of a redex is the degree of its w-abstraction. The* **max-degree** *of a term $t \in \Lambda^{\mathtt{im}}_{\cap}$ is written $\delta_{\mathsf{max}}(t)$ and defined as the maximum degree of the redexes in t, or 0 if t has no redexes. This notion is also defined for set-terms $(\delta_{\mathsf{max}}(\vec{t}))$ and lists of wrappers $(\delta_{\mathsf{max}}(\mathtt{L}))$ in a similar way. The* **weight** *of a term $t \in \Lambda^{\mathtt{im}}_{\cap}$ is written $\mathsf{w}(t)$ and defined as the number of wrappers in t.*

Example 3.2 *In order to illustrate how the system works, and, especially, how wrappers are handled, we show two possible reductions for a term involving self-application and reductions inside the argument.*

We underline the contracted redex and highlight the new introduced memory at each step for clarity. Let $A = a \to a$; $A^A = A \to A$; $I^a = \lambda x^{\{a\}}.\, x$; $I^A = \lambda x^{\{A\}}.\, x$; $I^{A^A} = \lambda x^{\{A^A\}}.\, x$.



Example 3.3 *The following reductions show how erased subterms are memorized by wrappers.*

1. $(\lambda x^{\{A\}}.\, y^B)((\lambda x^{\{B\}}.\, z^A) w^B) \quad \to_{\mathtt{im}} \quad y^B\langle \boldsymbol{(\lambda x^{\{B\}}.\, z^A) w^B}\rangle \quad \to_{\mathtt{im}} \quad y^B\langle z^A\langle \boldsymbol{w^B}\rangle\rangle$
2. $(\lambda x^{\{A\}}.\, \lambda y^{\{B\}}.\, x) z^A w^B \quad \to_{\mathtt{im}} \quad (\lambda y^{\{B\}}.\, z^A)\langle \boldsymbol{z^A}\rangle w^B \quad \to_{\mathtt{im}} \quad z^A\langle \boldsymbol{w^B}\rangle\langle z^A\rangle$

Remark 3.4 *Each step* $t \to_{\mathtt{i}} s$ *in the* $\Lambda^{\mathtt{i}}_{\cap}$*-calculus (without wrappers) has a* **corresponding** *step* $t \to_{\mathtt{im}} s'$ *in the* $\Lambda^{\mathtt{im}}_{\cap}$*-calculus (with wrappers), contracting the same redex but creating one wrapper. For example, the step* $(\lambda x.\, z\, x\, x)\, I \to_{\mathtt{i}} z\, I\, I$ *has a corresponding step* $(\lambda x.\, z\, x\, x)\, I \to_{\mathtt{im}} (z\, I\, I)\langle I\rangle$*. In particular an erasing reduction step in* $\Lambda^{\mathtt{i}}_{\cap}$ *is mapped a non-erasing one in* $\Lambda^{\mathtt{im}}_{\cap}$*. For example,* $(\lambda x^{\vec{A}}.\, z^B)\, y^{\vec{A}} \to_{\mathtt{i}} z^B$*, while* $(\lambda x^{\vec{A}}.\, z^B)\, y^{\vec{A}} \to_{\mathtt{im}} z^B\langle y^{\vec{A}}\rangle$.

The following properties hold for the $\Lambda^{\mathtt{im}}_{\cap}$ calculus. Subject expansion also holds for this calculus, but it is not needed as part of the technical development. See Appendices A.2.1 and A.2.2 for detailed proofs.

Proposition 3.5 (Subject reduction) *If* $t \to_{\mathtt{im}} t'$ *and* $\Gamma \vdash_{\mathtt{im}} t : A$*, then* $\Gamma \vdash_{\mathtt{im}} t' : A$.

Proposition 3.6 (Confluence) *If* $t_1 \to^*_{\mathtt{im}} t_2$ *and* $t_1 \to^*_{\mathtt{im}} t_3$*, there exists a term* t_4 *such that* $t_2 \to^*_{\mathtt{im}} t_4$ *and* $t_3 \to^*_{\mathtt{im}} t_4$.

3.2 Simplification by complete developments

We now introduce the operations of *simplification* and *full simplification* of a term. The *simplification* computes the result of the complete development of the redexes of a given degree, *i.e.*, reduces in parallel all the redexes of that degree. The *full simplification* iteratively applies the simplification of a term, starting with the maximum degree and decreasing down to degree 1. Both operations are well-defined by recursion: simplification on the structure of terms, and full simplification on the maximum degree of the term. Turing's observation (redex contractions cannot create redexes of higher or equal degree) is a key underlying property that ensures that full simplification works as intended, *i.e.*, it computes the normal form of a term. The subsequent lemmas show it.

Definition 3.7 (Simplification) *For each integer* $d \geq 1$ *we define an operation written* $\mathtt{S}_d(-)$*, called* **simplification of degree** *d, that can be applied on terms* ($\mathtt{S}_d(t)$)*, set-terms* ($\mathtt{S}_d(\vec{t})$)*, and lists of wrappers*

($\mathtt{S}_d(\mathtt{L})$), mutually recursively as follows:

$$
\begin{aligned}
\mathtt{S}_d(x^A) &:= x^A \\
\mathtt{S}_d(\lambda x^{\vec{A}}.\, t) &:= \lambda x^{\vec{A}}.\, \mathtt{S}_d(t) \\
\mathtt{S}_d(t\,\vec{s}) &:= \begin{cases} \mathtt{S}_d(t')[x^{\vec{A}} := \mathtt{S}_d(\vec{s})]\langle \mathtt{S}_d(\vec{s})\rangle \mathtt{S}_d(\mathtt{L}) & \text{if } t = (\lambda x^{\vec{A}}.\, t')\mathtt{L} \text{ of degree } d \\ \mathtt{S}_d(t)\,\mathtt{S}_d(\vec{s}) & \text{otherwise} \end{cases} \\
\mathtt{S}_d(t\langle \vec{s}\rangle) &:= \mathtt{S}_d(t)\langle \mathtt{S}_d(\vec{s})\rangle \\
\mathtt{S}_d(\{t_1, \ldots, t_n\}) &:= \{\mathtt{S}_d(t_1), \ldots, \mathtt{S}_d(t_n)\} \\
\mathtt{S}_d(\Box\langle \vec{t_1}\rangle \ldots \langle \vec{t_n}\rangle) &:= \Box\langle \mathtt{S}_d(\vec{t_1})\rangle \ldots \langle \mathtt{S}_d(\vec{t_n})\rangle
\end{aligned}
$$

If $t \in \Lambda_{\cap}^{\mathtt{im}}$ is a term and $D = \delta_{\mathsf{max}}(t)$, the **full simplification of** *t is written $\mathtt{S}_*(t)$ and defined as $\mathtt{S}_*(t) := \mathtt{S}_1(\ldots \mathtt{S}_{D-1}(\mathtt{S}_D(t))\ldots)$.*

Lemma 3.8 (Soundness of simplification) *If $t \in \Lambda_{\cap}^{\mathtt{im}}$ and $d \geq 1$ then $t \to_{\mathtt{im}}^* \mathtt{S}_d(t)$.*

Proof. Straightforward by mutual induction on the term t, set-term $\vec{t}$ and list of wrappers $\mathtt{L}$, generalizing the statement to set-terms and lists of wrappers, *i.e.* $\vec{t} \to_{\mathtt{im}}^* \mathtt{S}_d(\vec{t})$ and $\mathtt{L} \to_{\mathtt{im}}^* \mathtt{S}_d(\mathtt{L})$. □

Lemma 3.9 (Creation of abstraction by substitution) *Let $t \in \Lambda_{\cap}^{\mathtt{im}}$ and $\vec{s} \subset \Lambda_{\cap}^{\mathtt{im}}$ such that t is not a w-abstraction but $t[x^{\vec{A}} := \vec{s}]$ is a w-abstraction. Then $t = x^B\mathtt{L}$ with $B \in \vec{A}$.*

Proof. Let us write t as of the form $t = t'\mathtt{L}$ where t' is not a wrapper. Note that t' is not an abstraction, because by hypothesis t is not a w-abstraction. Moreover, t' cannot be an application, because if $t' = t_1\,t_2$ then $t[x^{\vec{A}} := \vec{s}] = t'[x^{\vec{A}} := \vec{s}](\mathtt{L}[x^{\vec{A}} := \vec{s}]) = (t_1[x^{\vec{A}} := \vec{s}]\,t_2[x^{\vec{A}} := \vec{s}])(\mathtt{L}[x^{\vec{A}} := \vec{s}])$ would not be a w-abstraction. The only remaining possibility is that t' is a variable, so $t' = y^B$. If it is a variable other than x, *i.e.* $y \neq x$, then $t[x^{\vec{A}} := \vec{s}] = t'[x^{\vec{A}} := \vec{s}](\mathtt{L}[x^{\vec{A}} := \vec{s}]) = y^B(\mathtt{L}[x^{\vec{A}} := \vec{s}])$ would not be a w-abstraction. So we necessarily have that $y = x^B$ with $B \in \vec{A}$, which concludes the proof. □

Lemma 3.10 (Bound for the max-degree of a substitution) *Let $\Gamma \Vdash_{\mathtt{im}} \vec{s} : \vec{A}$, where $\delta_{\mathsf{max}}(\vec{s}) < d$ and $\mathsf{h}(\vec{A}) < d$. If $\Gamma, x : \vec{A} \vdash_{\mathtt{im}} t : B$ and $\delta_{\mathsf{max}}(t) < d$ then $\delta_{\mathsf{max}}(t[x^{\vec{A}} := \vec{s}]) < d$.*

Proof. We generalize the statement for set-terms, claiming that $\Gamma, x : \vec{A} \Vdash_{\mathtt{im}} \vec{t} : \vec{B}$ and $\delta_{\mathsf{max}}(\vec{t}) < d$ imply $\delta_{\mathsf{max}}(\vec{t}[x^{\vec{A}} := \vec{s}]) < d$. The proof proceeds by simultaneous induction on the structure of the term t or set-term $\vec{t}$. The interesting cases are when the term is a variable or an application. The remaining cases are straightforward by resorting to the IH. If t is a **variable**, $t = y^C$, we consider two subcases, depending on whether $y = x$ or not.

1. If $x = y$, then $C = B$ and since $\Gamma, x : \vec{A} \vdash_{\mathtt{im}} x : B$ holds, we have that $B \in \vec{A}$. Hence there is a term $s_0 \in \vec{s}$ such that $\Gamma \vdash_{\mathtt{im}} s_0 : B$, and $x^B[x^{\vec{A}} := \vec{s}] = s_0$. In particular, by definition we have that $\delta_{\mathsf{max}}(s_0) \leq \delta_{\mathsf{max}}(\vec{s})$, and by hypothesis we have that $\delta_{\mathsf{max}}(\vec{s}) < d$, so $\delta_{\mathsf{max}}(t[x^{\vec{A}} := \vec{s}]) = \delta_{\mathsf{max}}(x^B[x^{\vec{A}} := \vec{s}]) = \delta_{\mathsf{max}}(s_0) < d$, as required.
2. If $x \neq y$, then $\delta_{\mathsf{max}}(t[x^{\vec{A}} := \vec{s}]) = \delta_{\mathsf{max}}(y^C[x^{\vec{A}} := \vec{s}]) = \delta_{\mathsf{max}}(y^C) = \delta_{\mathsf{max}}(t) < d$.

If t is an **application**, $t = u\,\vec{r}$, note that u is a subterm of t so $\delta_{\mathsf{max}}(u) \leq \delta_{\mathsf{max}}(t) < d$, and similarly $\delta_{\mathsf{max}}(\vec{r}) \leq \delta_{\mathsf{max}}(t) < d$. Thus applying the IH on each subterm we have that $\delta_{\mathsf{max}}(u[x^{\vec{A}} := \vec{s}]) < d$ and that $\delta_{\mathsf{max}}(\vec{r}[x^{\vec{A}} := \vec{s}]) < d$. We consider three subcases, depending on whether **1.** u is a w-abstraction, **2.** u is not a w-abstraction and $u[x^{\vec{A}} := \vec{s}]$ is a w-abstraction, or **3.** u and $u[x^{\vec{A}} := \vec{s}]$ are not w-abstractions:

1. If u is a w-abstraction: then $u = (\lambda y^{\vec{C}}.\, u')\mathtt{L}$ is of type $\vec{C} \to D$. Let k be the degree of the w-

abstraction u, *i.e.* the height of its type, $k = \mathsf{h}(\vec{C} \to D)$. By definition, $\delta_{\mathsf{max}}(t) = \delta_{\mathsf{max}}(u\,\vec{r}) = \max\{k, \delta_{\mathsf{max}}(u), \delta_{\mathsf{max}}(\vec{r})\}$. Since $\delta_{\mathsf{max}}(t) < d$ by hypothesis, in particular we have that $k < d$. Moreover, note that $u[x^{\vec{A}} := \vec{s}] = (\lambda y^{\vec{C}}.\, u'[x^{\vec{A}} := \vec{s}])(\mathtt{L}[x^{\vec{A}} := \vec{s}])$, so $u[x^{\vec{A}} := \vec{s}]$ is a w-abstraction, and it is of degree k because substitution preserves the type of a term. Therefore:

$$\begin{aligned}\delta_{\mathsf{max}}(t[x^{\vec{A}} := \vec{s}]) &= \delta_{\mathsf{max}}(u[x^{\vec{A}} := \vec{s}]\,\vec{r}[x^{\vec{A}} := \vec{s}]) \\ &= \max\{k, \delta_{\mathsf{max}}(u[x^{\vec{A}} := \vec{s}]), \delta_{\mathsf{max}}(\vec{r}[x^{\vec{A}} := \vec{s}])\} < d\end{aligned}$$

The last step is justified because we have already noted that $k < d$ and $\delta_{\mathsf{max}}(u[x^{\vec{A}} := \vec{s}]) < d$ and $\delta_{\mathsf{max}}(\vec{r}[x^{\vec{A}} := \vec{s}]) < d$.

2. If u is not a w-abstraction and $u[x^{\vec{A}} := \vec{s}]$ is a w-abstraction: then by Lemma 3.9 u must be of the form $x^C\mathtt{L}$ with $A \in C$. Let k be the degree of the w-abstraction $u[x^{\vec{A}} := \vec{s}]$. Then k is the height of the type of $u[x^{\vec{A}} := \vec{s}]$, that is, $k = \mathsf{h}(C) \leq \mathsf{h}(\vec{A}) < d$. To conclude, note that:

$$\begin{aligned}\delta_{\mathsf{max}}(t[x^{\vec{A}} := \vec{s}]) &= \delta_{\mathsf{max}}(u[x^{\vec{A}} := \vec{s}]\,\vec{r}[x^{\vec{A}} := \vec{s}]) \\ &= \max\{k, \delta_{\mathsf{max}}(u[x^{\vec{A}} := \vec{s}]), \delta_{\mathsf{max}}(\vec{r}[x^{\vec{A}} := \vec{s}])\} < d\end{aligned}$$

The last step is justified because we have already noted that $k < d$ and $\delta_{\mathsf{max}}(u[x^{\vec{A}} := \vec{s}]) < d$ and $\delta_{\mathsf{max}}(\vec{r}[x^{\vec{A}} := \vec{s}]) < d$.

3. If u and $u[x^{\vec{A}} := \vec{s}]$ are not a w-abstractions: then

$$\begin{aligned}\delta_{\mathsf{max}}(t[x^{\vec{A}} := \vec{s}]) &= \delta_{\mathsf{max}}(u[x^{\vec{A}} := \vec{s}]\,\vec{r}[x^{\vec{A}} := \vec{s}]) \\ &= \max\{\delta_{\mathsf{max}}(u[x^{\vec{A}} := \vec{s}]\,), \delta_{\mathsf{max}}(\vec{r}[x^{\vec{A}} := \vec{s}])\} < d\end{aligned}$$

The last step is justified because we have already noted that $\delta_{\mathsf{max}}(u[x^{\vec{A}} := \vec{s}]) < d$ and $\delta_{\mathsf{max}}(\vec{r}[x^{\vec{A}} := \vec{s}]) < d$.

□

Lemma 3.11 (Simplification does not create abstractions) *Suppose that $t \in \Lambda^{\mathtt{im}}_{\cap}$ is not a w-abstraction and $\delta_{\mathsf{max}}(t) \leq d$ and $\mathsf{h}(\mathsf{type}(t)) \geq d$. Then $\mathtt{S}_d(t)$ is not a w-abstraction.*

Proof. We proceed by induction on t. The interesting case is when t is a redex of degree d. We claim that this case is impossible. Indeed, suppose that $t = (\lambda x^{\vec{A}}.\, s)\mathtt{L}\,\vec{u}$, where $\lambda x^{\vec{A}}.\, s$ is of type $\vec{A} \to B$ so that t is of type B. The degree of the redex is $d = \mathsf{h}(\vec{A} \to B)$ and $\mathsf{h}(\mathsf{type}(t)) = \mathsf{h}(B) < d$. However, by hypothesis, $\mathsf{h}(\mathsf{type}(t)) \geq d$, from which we obtain a contradiction. The remaining cases are straightforward resorting to the IH. □

Lemma 3.12 (Simplification decreases the max-degree) *If $d \geq 1$ and $\delta_{\mathsf{max}}(t) \leq d$ then $\delta_{\mathsf{max}}(\mathtt{S}_d(t)) < d$.*

Proof. We proceed by simultaneous induction, generalizing the statement for set-terms ($\delta_{\mathsf{max}}(\vec{t}) \leq d$ implies $\delta_{\mathsf{max}}(\mathtt{S}_d(\vec{t})) < d$) and lists of wrappers ($\delta_{\mathsf{max}}(\mathtt{L}) \leq d$ implies $\delta_{\mathsf{max}}(\mathtt{S}_d(\mathtt{L})) < d$). The interesting cases are when t is a variable or an application. The remaining cases are straightforward by IH. If t is a **variable**, $t = x^A$, then $\delta_{\mathsf{max}}(\mathtt{S}_d(x^A)) = \delta_{\mathsf{max}}(x^A) = 0 < d$ because $d \geq 1$ by hypothesis. If t is an **application**, $t = s\,\vec{u}$, we consider two cases, depending on whether s is a w-abstraction of degree d or not:

1. If $s = (\lambda x^{\vec{A}}.\, s')\mathtt{L}$ is a w-abstraction of degree d: then note that s' is a subterm of t which implies that $\delta_{\mathsf{max}}(s') \leq \delta_{\mathsf{max}}(t) \leq d$, and this in turn implies by IH that $\delta_{\mathsf{max}}(\mathtt{S}_d(s')) < d$. Similarly, we can note

that $\delta_{\mathsf{max}}(\mathtt{L}) \leq d$ so by IH $\delta_{\mathsf{max}}(\mathtt{S}_d(\mathtt{L})) < d$, and that $\delta_{\mathsf{max}}(\vec{u}) \leq d$ so by IH $\delta_{\mathsf{max}}(\mathtt{S}_d(\vec{u})) < d$. Since the term is well-typed, the type of the w-abstraction $s = (\lambda x^{\vec{A}}.\, s')\mathtt{L}$ is of the form $\vec{A} \to B$, while the type of $\vec{u}$ is $\vec{A}$. Moreover, $\mathsf{h}(\vec{A} \to B) = d$ by hypothesis, so in particular $\mathsf{h}(\vec{A}) < d$. From this we obtain by Lemma 3.10 that $\delta_{\mathsf{max}}(\mathtt{S}_d(s')[x^{\vec{A}} := \mathtt{S}_d(\vec{u})]) < d$. Finally, we have:

$$\begin{aligned}\delta_{\mathsf{max}}(\mathtt{S}_d(s\,\vec{u})) &= \delta_{\mathsf{max}}(\mathtt{S}_d(s')[x^{\vec{A}} := \mathtt{S}_d(\vec{u})]\langle \mathtt{S}_d(\vec{u})\rangle \mathtt{S}_d(\mathtt{L})) \\ &= \max\{\delta_{\mathsf{max}}(\mathtt{S}_d(s')[x^{\vec{A}} := \mathtt{S}_d(\vec{u})]), \delta_{\mathsf{max}}(\mathtt{S}_d(\vec{u})), \delta_{\mathsf{max}}(\mathtt{S}_d(\mathtt{L}))\} < d\end{aligned}$$

2. If s is not a w-abstraction of degree d: since s is a subterm of t we have that $\delta_{\mathsf{max}}(s) \leq \delta_{\mathsf{max}}(t) \leq d$ so by IH $\delta_{\mathsf{max}}(\mathtt{S}_d(s)) < d$. Similarly, $\delta_{\mathsf{max}}(\vec{u}) \leq \delta_{\mathsf{max}}(t) \leq d$ so by IH $\delta_{\mathsf{max}}(\mathtt{S}_d(\vec{u})) < d$. Let the type of s be of the form $\vec{A} \to B$ and let $k := \mathsf{h}(\vec{A} \to B)$. We consider two cases, depending on whether $k < d$ or $k \geq d$.

   If $k = \mathsf{h}(\vec{A} \to B) < d$: then by Lemma 3.8 we have $s \to_{\mathtt{im}}^* \mathtt{S}_d(s)$ and by Subject Reduction (3.5) the type of $\mathtt{S}_d(s)$ is also $\vec{A} \to B$. Hence:

$$\delta_{\mathsf{max}}(\mathtt{S}_d(t)) = \delta_{\mathsf{max}}(\mathtt{S}_d(s\,\vec{u})) = \delta_{\mathsf{max}}(\mathtt{S}_d(s)\,\mathtt{S}_d(\vec{u})) = \max\{k, \delta_{\mathsf{max}}(s), \delta_{\mathsf{max}}(\vec{u})\} < d$$

   On the other hand, if $k = \mathsf{h}(\vec{A} \to B) \geq d$, we claim that s cannot be a w-abstraction. First, note by hypothesis that s is not a w-abstraction of degree $k = d$. Second, s cannot be a w-abstraction of degree $k > d$, as we would have that $s\,\vec{u}$ is a redex of degree k, so $\delta_{\mathsf{max}}(s\,\vec{u}) \geq k$. This would imply that $d < k \leq \delta_{\mathsf{max}}(s\,\vec{u}) \leq d$, a contradiction. Since s is not a w-abstraction, its type is $\vec{A} \to B$ of height $k \geq d$, and $\delta_{\mathsf{max}}(s) \leq d$, by Lemma 3.11 we have that $\mathtt{S}_d(s)$ is not a w-abstraction. Hence $\mathtt{S}_d(s)\,\mathtt{S}_d(\vec{u})$ is not a redex, and $\delta_{\mathsf{max}}(\mathtt{S}_d(t)) = \delta_{\mathsf{max}}(\mathtt{S}_d(s\,\vec{u})) = \delta_{\mathsf{max}}(\mathtt{S}_d(s)\,\mathtt{S}_d(\vec{u})) = \max\{\delta_{\mathsf{max}}(\mathtt{S}_d(s)), \delta_{\mathsf{max}}(\mathtt{S}_d(\vec{u}))\} < d$.

□

Proposition 3.13 (Full simplification yields the normal form) *Let $t \in \Lambda_\cap^{\mathtt{im}}$. Then $t \to_{\mathtt{im}}^* \mathtt{S}_*(t)$ and $\mathtt{S}_*(t)$ is in $\to_{\mathtt{im}}$-normal form.*

Proof. Let d be the max-degree of t and define $\mathtt{S}_{>k}(t) := \mathtt{S}_{k+1}(\ldots \mathtt{S}_{d-1}(\mathtt{S}_d(t))\ldots)$ for each k such that $0 \leq k \leq d$. Note that $\mathtt{S}_{>d}(t) = t$ and $\mathtt{S}_{>0}(t) = \mathtt{S}_*(t)$. To show that $t \to_{\mathtt{im}}^* \mathtt{S}_*(t)$, note that $\mathtt{S}_{>k}(t) \to_{\mathtt{im}}^* \mathtt{S}_k(\mathtt{S}_{>k}(t)) = \mathtt{S}_{>k-1}(t)$ for each $1 \leq k \leq d$ by Lemma 3.8, so:

$$t = \mathtt{S}_{>d}(t) \to_{\mathtt{im}}^* \mathtt{S}_{>d-1}(t) \to_{\mathtt{im}}^* \ldots \to_{\mathtt{im}}^* \mathtt{S}_{>k}(t) \to_{\mathtt{im}}^* \ldots \to_{\mathtt{im}}^* \mathtt{S}_{>0}(t) = \mathtt{S}_*(t)$$

To show that $\mathtt{S}_*(t)$ is a $\to_{\mathtt{im}}$-normal form, we claim that for each $0 \leq k \leq d$ we have that $\delta_{\mathsf{max}}(\mathtt{S}_{>k}(t)) \leq k$. We proceed by induction on $d - k$. If $d - k = 0$, we have that $k = d$, so $\delta_{\mathsf{max}}(\mathtt{S}_{>d}(t)) = \delta_{\mathsf{max}}(t) = d$, since d is the max-degree of t. If $d - k > 0$, we have that $0 \leq k < d$. By IH, $\delta_{\mathsf{max}}(\mathtt{S}_{>k+1}(t)) \leq k + 1$. Then $\delta_{\mathsf{max}}(\mathtt{S}_{>k}(t)) = \delta_{\mathsf{max}}(\mathtt{S}_{k+1}(\mathtt{S}_{>k+1}(t))) < k + 1$ by Lemma 3.12. This means that $\delta_{\mathsf{max}}(\mathtt{S}_{>k}(t)) \leq k$, as required. □

3.3 The decreasing measure

The $\mathcal{W}$-measure for $\Lambda_\cap^{\mathtt{i}}$ is defined in two steps: computing the normal form in $\Lambda_\cap^{\mathtt{im}}$, and counting the number of wrappers in it. The first step was taken care of in the previous section. The second one is simple, it just requires to analyze the resulting term of the full simplification.

The core of this section is proving that the measure decreases, *i.e.*, $t \to_{\mathtt{i}} s$ implies $\mathcal{W}(t) > \mathcal{W}(s)$. The proof is carried out by observing how $\to_{\mathtt{i}}$ and $\to_{\mathtt{im}}$ differ, and how wrappers in $\Lambda_\cap^{\mathtt{im}}$ are handled. Confluence holds in $\Lambda_\cap^{\mathtt{im}}$, so terms share normal form, and therefore full simplification, with all their reducts. The intuition behind the proof is that, if we have $t \to_{\mathtt{i}} s$, then the corresponding step $t \to_{\mathtt{im}} s'$ is responsible for the presence of at least one wrapper in $\mathtt{S}_*(t)$ that is absent from $\mathtt{S}_*(s)$.

Definition 3.14 (Forgetful reduction) *We define a binary relation $t \rhd s$ between typable terms as the closure by congruence under arbitrary contexts of the axiom $t\langle\vec{s}\rangle \rhd t$.*

Remark 3.15 *If t is in $\to_{\mathtt{im}}$-normal form and $t \rhd s$ then s is also in $\to_{\mathtt{im}}$-normal form.*

Remark 3.16 *If $t \rhd s$ then $\mathsf{w}(t) > \mathsf{w}(s)$.*

Lemma 3.17 (Reduce/forget lemma) *Let $t \to_{\mathtt{i}} s$ be a step in the $\Lambda^{\mathtt{i}}_{\cap}$-calculus (without wrappers) and consider its corresponding step $t \to_{\mathtt{im}} s'$. Then $s' \rhd s$.*

Proof. Straightforward by induction on t. □

Lemma 3.18 (Forgetful reduction commutes with reduction) *If $t_1 \to^*_{\mathtt{im}} t_2$ and $t_1 \rhd^+ t_3$ there exists t_4 such that $t_2 \rhd^+ t_4$ and $t_3 \to^=_{\mathtt{im}} t_4$.*

Proof. It suffices to prove a local commutation result, namely that if $t_1 \to_{\mathtt{im}} t_2$ and $t_1 \rhd t_3$ there exists t_4 such that $t_2 \rhd^+ t_4$ and $t_3 \to^=_{\mathtt{im}} t_4$. Graphically:

$$\begin{array}{ccc} t_1 & \to_{\mathtt{im}} & t_2 \\ \triangledown & & \triangledown^+ \\ t_3 & \to^=_{\mathtt{im}} & t_4 \end{array}$$

We proceed by induction on t_1.

1. $t_1 = x^{\vec{A}}$: this case is impossible, since x does not reduce.
2. $t_1 = \lambda x^{\vec{A}}.\, t_1'$: then $t_2 = \lambda x^{\vec{A}}.\, t_2'$ and $t_3 = \lambda x^{\vec{A}}.\, t_3'$, with $t_1' \to_{\mathtt{im}} t_2'$ and $t_1' \rhd t_3'$. By IH there exists t_4' s.t. $t_3' \to^=_{\mathtt{im}} t_4'$ and $t_2' \rhd^+ t_4'$. By congruence, the following holds

$$\begin{array}{ccc} \lambda x^{\vec{A}}.\, t_1' & \to_{\mathtt{im}} & \lambda x^{\vec{A}}.\, t_2' \\ \triangledown & & \triangledown^+ \\ \lambda x^{\vec{A}}.\, t_3' & \to^=_{\mathtt{im}} & \lambda x^{\vec{A}}.\, t_4' \end{array}$$

3. $t_1 = t_{11}\,\vec{t}_{12}$. We consider two subcases here, depending on t_{11} being a w-abstraction and on which redex is contracted.
   3.1 $t_1 = (\lambda x^{\vec{A}}.\, t_{11})\mathtt{L}\,\vec{t}_{12}$ and the contracted redex is at head position. Then

$$\begin{array}{ccc} (\lambda x^{\vec{A}}.\, t_{11})\mathtt{L}\,\vec{t}_{12} & \to_{\mathtt{im}} & t_{11}[x^{\vec{A}} := \vec{t}_{12}]\langle\vec{t}_{12}\rangle\mathtt{L} \\ \triangledown & & \\ (\lambda x^{\vec{A}}.\, t_{11}')\mathtt{L}'\,\vec{t}_{12}' & & \end{array}$$

   where since there is a single $\rhd$-step, only one of t_{11}', $\mathtt{L}'$, or $\vec{t}_{12}'$, can be different from t_{11}, $\mathtt{L}$, or $\vec{t}_{12}$, respectively. If the $\rhd$-step occurs in t_{11} or $\mathtt{L}$, then the same step can be reproduced from t_2. Else, considering $\vec{t}_{12}$ can appear more than once in t_2, the $\rhd$-step may have to be reproduced several times, hence the $\rhd^+$. Then we have that

$$\begin{array}{ccc} (\lambda x^{\vec{A}}.\, t_{11})\mathtt{L}\,\vec{t}_{12} & \to_{\mathtt{im}} & t_{11}[x^{\vec{A}} := \vec{t}_{12}]\langle\vec{t}_{12}\rangle\mathtt{L} \\ \triangledown & & \triangledown^+ \\ (\lambda x^{\vec{A}}.\, t_{11}')\mathtt{L}'\,\vec{t}_{12}' & \to_{\mathtt{im}} & t_{11}'[x^{\vec{A}} := \vec{t}_{12}']\langle\vec{t}_{12}'\rangle\mathtt{L}' \end{array}$$

   3.2 t_{11} is not a w-abstraction or the contracted redex is not the one at head position. We consider two subcases.

3.2.1 Both steps occur in the same subterm. It suffices to resort to the IH:

$$\begin{array}{ccc} t_{11}\vec{t}_{12} & \to_{\mathtt{im}} & t'_{11}\vec{t}_{12} \\ \triangledown & & \triangledown^{+} \\ t''_{11}\vec{t}_{12} & \to^{=}_{\mathtt{im}} & t'''_{11}\vec{t}_{12} \end{array} \qquad \begin{array}{ccc} t_{11}\vec{t}_{12} & \to_{\mathtt{im}} & t_{11}\vec{t}'_{12} \\ \triangledown & & \triangledown^{+} \\ t_{11}\vec{t}'_{12} & \to^{=}_{\mathtt{im}} & t_{11}\vec{t}''_{12} \end{array}$$

3.2.2 The steps occur in different subterms. Then the diagram can be closed immediately:

$$\begin{array}{ccc} t_{11}\vec{t}_{12} & \to_{\mathtt{im}} & t'_{11}\vec{t}_{12} \\ \triangledown & & \triangledown \\ t_{11}\vec{t}'_{12} & \to_{\mathtt{im}} & t'_{11}\vec{t}'_{12} \end{array} \qquad \begin{array}{ccc} t_{11}\vec{t}_{12} & \to_{\mathtt{im}} & t_{11}\vec{t}'_{12} \\ \triangledown & & \triangledown \\ t'_{11}\vec{t}_{12} & \to_{\mathtt{im}} & t'_{11}\vec{t}'_{12} \end{array}$$

4. $t_1 = t_{11}\langle \vec{t}_{12} \rangle$. We consider three general subcases:

4.1 Both steps occur in the same subterm. It suffices to resort to the IH:

$$\begin{array}{ccc} t_{11}\langle \vec{t}_{12} \rangle & \to_{\mathtt{im}} & t'_{11}\langle \vec{t}_{12} \rangle \\ \triangledown & & \triangledown^{+} \\ t''_{11}\langle \vec{t}_{12} \rangle & \to^{=}_{\mathtt{im}} & t'''_{11}\langle \vec{t}_{12} \rangle \end{array} \qquad \begin{array}{ccc} t_{11}\langle \vec{t}_{12} \rangle & \to_{\mathtt{im}} & t_{11}\langle \vec{t}'_{12} \rangle \\ \triangledown & & \triangledown^{+} \\ t_{11}\langle \vec{t}'_{12} \rangle & \to^{=}_{\mathtt{im}} & t_{11}\langle \vec{t}''_{12} \rangle \end{array}$$

4.2 The steps occur in different subterms. Then the diagram can be closed immediately:

$$\begin{array}{ccc} t_{11}\langle \vec{t}_{12} \rangle & \to_{\mathtt{im}} & t'_{11}\langle \vec{t}_{12} \rangle \\ \triangledown & & \triangledown \\ t_{11}\langle \vec{t}'_{12} \rangle & \to_{\mathtt{im}} & t'_{11}\langle \vec{t}'_{12} \rangle \end{array} \qquad \begin{array}{ccc} t_{11}\langle \vec{t}_{12} \rangle & \to_{\mathtt{im}} & t_{11}\langle \vec{t}'_{12} \rangle \\ \triangledown & & \triangledown \\ t'_{11}\langle \vec{t}_{12} \rangle & \to_{\mathtt{im}} & t'_{11}\langle \vec{t}'_{12} \rangle \end{array}$$

4.3 The $\triangleright$-step occur at the root. We perform the $\to_{\mathtt{im}}$ step, or not.

$$\begin{array}{ccc} t_{11}\langle \vec{t}_{12} \rangle & \to_{\mathtt{im}} & t'_{11}\langle \vec{t}_{12} \rangle \\ \triangledown & & \triangledown \\ t_{11} & \to_{\mathtt{im}} & t'_{11} \end{array} \qquad \begin{array}{ccc} t_{11}\langle \vec{t}_{12} \rangle & \to_{\mathtt{im}} & t_{11}\langle \vec{t}'_{12} \rangle \\ \triangledown & & \triangledown \\ t_{11} & = & t_{11} \end{array}$$

5. $t_1 = \{s_1, \ldots, s_n\}$. We consider two subcases.

5.1 Both steps occur in the same subterm. It suffices to resort to the IH:

$$\begin{array}{ccc} \{s_1, \ldots, s_i, \ldots, s_n\} & \to_{\mathtt{im}} & \{s_1, \ldots, s'_i, \ldots, s_n\} \\ \triangledown & & \triangledown^{+} \\ \{s_1, \ldots, s''_i, \ldots, s_n\} & \to^{=}_{\mathtt{im}} & \{s_1, \ldots, s'''_i, \ldots, s_n\} \end{array}$$

5.2 The steps occur in different subterms. Then the diagram can be closed immediately. Indeed, let $i, j \in 1..n$:

$$\begin{array}{ccc} \{s_1, \ldots, s_i, \ldots, s_j, \ldots, s_n\} & \to_{\mathtt{im}} & \{s_1, \ldots, s'_i, \ldots, s_j, \ldots, s_n\} \\ \triangledown & & \triangledown \\ \{s_1, \ldots, s_i, \ldots, s'_j, \ldots, s_n\} & \to_{\mathtt{im}} & \{s_1, \ldots, s'_i, \ldots, s'_j, \ldots, s_n\} \end{array}$$

□

Lemma 3.19 (Properties of the full simplification)
1. *If* $t \to_{\mathtt{im}} s$ *then* $\mathtt{S}_*(t) = \mathtt{S}_*(s)$. **2.** *If* $t \rhd s$ *then* $\mathtt{S}_*(t) \rhd^{+} \mathtt{S}_*(s)$.

Proof. We prove each item separately. First, suppose that $t \to_{\mathtt{im}} s$. By the fact that full simplification yields the normal form of a term (Proposition 3.13), we know that $t \to^{*}_{\mathtt{im}} \mathtt{S}_*(t)$ and $s \to^{*}_{\mathtt{im}} \mathtt{S}_*(s)$, where both $\mathtt{S}_*(t)$ and $\mathtt{S}_*(s)$ are in $\to_{\mathtt{im}}$-normal form. By Confluence (Proposition 3.6) we conclude that $\mathtt{S}_*(t) = \mathtt{S}_*(s)$.

Second, suppose that $t \rhd s$. By the fact that full simplification yields the normal form of a term (Proposition 3.13), we know that $t \to^{*}_{\mathtt{im}} \mathtt{S}_*(t)$, where $\mathtt{S}_*(t)$ is in $\to_{\mathtt{im}}$-normal form. Since forgetful reduction commutes with reduction (Lemma 3.18), we have that there exists a term $u \in \Lambda^{\mathtt{im}}_{\cap}$ such that $s \to^{*}_{\mathtt{im}} u$ and $\mathtt{S}_*(t) \rhd^{+} u$. By Remark 3.15, since $\mathtt{S}_*(t)$ is in $\to_{\mathtt{im}}$-normal form, u is also in $\to_{\mathtt{im}}$-normal form. Moreover,

$s \to_{\mathtt{im}}^{*} \mathtt{S}_*(s)$ where $\mathtt{S}_*(s)$ is in $\to_{\mathtt{im}}$-normal form by Proposition 3.13. So by Confluence (Proposition 3.6) we have that $\mathtt{S}_*(s) = u$. Then we have that $\mathtt{S}_*(t) \rhd^+ u = \mathtt{S}_*(s)$, as required. □

Definition 3.20 (The $\mathcal{W}$-measure) *The measure of a term $t \in \Lambda^{\mathtt{i}}_{\cap}$ is defined as the weight of its full simplification, where we recall that the weight is the number of wrappers in a term:*

$$\mathcal{W}(t) := \mathsf{w}(\mathtt{S}_*(t))$$

Theorem 3.21 (The $\mathcal{W}$-measure is decreasing) *Let $t, s \in \Lambda^{\mathtt{i}}_{\cap}$ be typable terms of the $\Lambda^{\mathtt{i}}_{\cap}$-calculus (without wrappers). If $t \to_{\mathtt{i}} s$ then $\mathcal{W}(t) > \mathcal{W}(s)$.*

Proof. Let $t \to_{\mathtt{i}} s$ and consider the corresponding step $t \to_{\mathtt{im}} s'$. By the reduce/forget lemma (3.17) we have that $s' \rhd s$. By Lemma 3.19 (**1**) we have that $\mathtt{S}_*(t) = \mathtt{S}_*(s')$. By Lemma 3.19 (**2**) we have that $\mathtt{S}_*(s') \rhd^+ \mathtt{S}_*(s)$. Hence $\mathtt{S}_*(t) \rhd^+ \mathtt{S}_*(s)$. By Remark 3.16 we conclude that $\mathcal{W}(t) = \mathsf{w}(\mathtt{S}_*(t)) > \mathsf{w}(\mathtt{S}_*(s)) = \mathcal{W}(s)$. □

Corollary 3.22 *$\Lambda^{\mathtt{i}}_{\cap}$ is strongly normalizing.*

4 Conclusion

In this section we provide a proof of the fact that $\Lambda^{\mathtt{e}}_{\cap}$ characterizes strong normalization, based on the correspondence between it and $\Lambda^{\mathtt{i}}_{\cap}$. This proof justifies the design of $\Lambda^{\mathtt{e}}_{\cap}$, showing that it is a variant of the original system in [12], w.r.t. the typability power. Note that we make use of $\Lambda^{\mathtt{i}}_{\cap}$ to prove the second item for uniformity purposes; it can also be proved entirely within $\Lambda^{\mathtt{e}}_{\cap}$.

Corollary 4.1 *$\Lambda^{\mathtt{e}}_{\cap}$ characterizes strong normalization.*

Proof.

- Let $M \in \Lambda$ be a λ-term typable in the $\Lambda^{\mathtt{e}}_{\cap}$ system, *i.e.* such that $\Gamma \vdash_{\mathtt{e}} M : A$. By the Correspondence Theorem (2.24.1), there is a term $t \in \Lambda^{\mathtt{i}}_{\cap}$ typable in the $\Lambda^{\mathtt{i}}_{\cap}$ system such that $\Gamma \vdash_{\mathtt{i}} t : A$ and $t \sqsubset M$. By Corollary 3.22, t is strongly normalizing. From the Simulation Theorem (2.26), we can observe that an infinite reduction sequence $M \to_\beta M_1 \to_\beta M_2 \to_\beta \ldots$ would induce an infinite reduction sequence $t \to_{\mathtt{i}}^+ t_1 \to_{\mathtt{i}}^+ t_2 \to_{\mathtt{i}}^+ \ldots$, and this would contradict strong normalization of $\Lambda^{\mathtt{i}}_{\cap}$. Hence M is strongly normalizing.
- Let $M \in \Lambda$ be strongly normalizing. By Lemma 2.29, there is $t \in \Lambda^{\mathtt{i}}_{\cap}$ such that $t \sqsubset M$ and $\Gamma \vdash_{\mathtt{i}} t : A$, for some Γ and A. Then t is uniform, and, by Theorem 2.24.2, $\Gamma \vdash_{\mathtt{i}} t : A$ implies $\Gamma \vdash_{\mathtt{e}} t^{\mathtt{e}} : A$. Since $t^{\mathtt{e}} = M$, the proof is given.

□

As we said in Section 1, this is not a completely new result. The novelty of our work comes from the technique we propose and the simplicity of its output. The strong normalization property for idempotent intersection type systems has already been proved in various papers using both semantical and syntactical approaches, and the fact that $\Lambda^{\mathtt{e}}_{\cap}$ is a variant of them can be considered folklore. The semantical approach relies on arguments like computability or reducibility candidates (*e.g.* [29,5]), while the syntactical ones are based on a measure that decreases with the β-reduction; as far as we know, there are three other syntactical proofs [22,9,11]. Here we supply a further syntactical proof, which uses as key ingredient a Church version of idempotent intersection types that has good proof-theoretical properties. The notion of measure obtained is simpler than [22,9], being a natural number, and operates directly within the intersection system, allowing for refinement where measures based on the translation in [11] are constrained.

In the future, we would like to explore the possibility of providing an exact decreasing measure, *i.e.* a measure whose result is an exact upper bound for the length of the longest reduction chain starting from the term. While this refinement can be easily done for simple types, through a modification of the method in [2], the extension to intersection types presents challenges. In this case, the number of reduction steps

in the derivation tree in general is bigger than the number of β-reductions in the term, so it would be necessary to change the operational behaviour of $\Lambda^{\mathtt{i}}_{\cap}$.

Moreover, it would be interesting to study if the techniques given in this paper can be adapted to intersection type systems that characterize solvable terms, such as that of [14], for which no syntactic proofs of normalization have been proposed, as far as we know.

References

[1] Barbanera, F., M. Dezani-Ciancaglini, U. de'Liguoro and B. Venneri, *YACC: yet another church calculus - A birthday present for herman inspired by his supervisor activity*, in: V. Capretta, R. Krebbers and F. Wiedijk, editors, *Logics and Type Systems in Theory and Practice - Essays Dedicated to Herman Geuvers on The Occasion of His 60th Birthday*, volume 14560 of *Lecture Notes in Computer Science*, pages 17–35, Springer (2024).
https://doi.org/10.1007/978-3-031-61716-4_2

[2] Barenbaum, P. and C. Sottile, *Two decreasing measures for simply typed λ-terms*, in: M. Gaboardi and F. van Raamsdonk, editors, *8th International Conference on Formal Structures for Computation and Deduction, FSCD 2023, July 3-6, 2023, Rome, Italy*, volume 260 of *LIPIcs*, pages 11:1–11:19, Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2023).
https://doi.org/10.4230/LIPICS.FSCD.2023.11

[3] Barendregt, H., M. Coppo and M. Dezani-Ciancaglini, *A filter lambda model and the completeness of type assignment*, J. Symb. Log. **48**, pages 931–940 (1983).
https://doi.org/10.2307/2273659

[4] Barendregt, H. P., *The Lambda Calculus: Its Syntax and Semantics*, volume 103 of *Studies in Logic and the Foundations of Mathematics*, North-Holland, Amsterdam (1984).
https://doi.org/10.1016/c2009-0-14341-6

[5] Barendregt, H. P., W. Dekkers and R. Statman, *Lambda Calculus with Types*, Lambda Calculus with Types, Cambridge University Press (2013), ISBN 9780521766142.
https://doi.org/10.1017/CBO9781139032636

[6] Barendregt, H. P. and G. Manzonetto, *Turing's contributions to lambda calculus*, in: B. Cooper and J. van Leeuwen, editors, *Alan Turing - His Work and Impact*, pages 139–143, Elsevier (2013).
https://doi.org/10.1016/C2010-0-66380-2

[7] Bernadet, A. and S. Lengrand, *Complexity of strongly normalising lambda-terms via non-idempotent intersection types*, in: *FOSSACS 2011*, pages 88–107 (2011).
https://doi.org/10.1007/978-3-642-19805-2_7

[8] Bono, V., B. Venneri and L. Bettini, *A typed lambda calculus with intersection types*, Theoretical Computer Science **398**, pages 95–113 (2008).
https://doi.org/10.1016/j.tcs.2008.01.046

[9] Boudol, G., *On strong normalization in the intersection type discipline*, in: M. Hofmann, editor, *Typed Lambda Calculi and Applications, 6th International Conference, TLCA 2003, Valencia, Spain, June 10-12, 2003, Proceedings*, volume 2701 of *Lecture Notes in Computer Science*, pages 60–74, Springer (2003).
https://doi.org/10.1007/3-540-44904-3_5

[10] Bucciarelli, A., D. Kesner and D. Ventura, *Strong normalization through intersection types and memory*, in: M. R. F. Benevides and R. Thiemann, editors, *Proceedings of the Tenth Workshop on Logical and Semantic Frameworks, with Applications, LSFA 2015, Natal, Brazil, August 31 - September 1, 2015*, volume 323 of *Electronic Notes in Theoretical Computer Science*, pages 75–91, Elsevier (2015).
https://doi.org/10.1016/J.ENTCS.2016.06.006

[11] Bucciarelli, A., A. Piperno and I. Salvo, *Intersection types and lambda-definability*, Math. Struct. Comput. Sci. **13**, pages 15–53 (2003).
https://doi.org/10.1017/S0960129502003833

[12] Coppo, M. and M. Dezani-Ciancaglini, *An extension of the basic functionality theory for the λ-calculus*, Notre Dame J. Formal Log. **21**, pages 685–693 (1980).
https://doi.org/10.1305/NDJFL/1093883253

[13] Coppo, M., M. Dezani-Ciancaglini and P. Sallé, *Functional characterization of some semantic equalities inside lambda-calculus*, in: H. A. Maurer, editor, *Automata, Languages and Programming, 6th Colloquium, Graz, Austria, July 16-20, 1979, Proceedings*, volume 71 of *Lecture Notes in Computer Science*, pages 133–146, Springer (1979).
https://doi.org/10.1007/3-540-09510-1_11

[14] Coppo, M., M. Dezani-Ciancaglini and B. Venneri, *Functional characters of solvable terms*, Math. Log. Q. **27**, pages 45–58 (1981).
https://doi.org/10.1002/MALQ.19810270205

[15] de Carvalho, D., *Execution time of λ-terms via denotational semantics and intersection types*, Math. Struct. Comput. Sci. **28**, pages 1169–1203 (2018).
https://doi.org/10.1017/S0960129516000396

[16] de Groote, P., *The conservation theorem revisited*, in: M. Bezem and J. F. Groote, editors, *Typed Lambda Calculi and Applications, International Conference on Typed Lambda Calculi and Applications, TLCA '93, Utrecht, The Netherlands, March 16-18, 1993, Proceedings*, volume 664 of *Lecture Notes in Computer Science*, pages 163–178, Springer (1993).
https://doi.org/10.1007/BFb0037105

[17] de Vrijer, R., *Exactly estimating functionals and strong normalization*, in: *Indagationes Mathematicae (Proceedings)*, volume 90, pages 479–493, North-Holland (1987).
https://doi.org/10.1016/1385-7258(87)90012-6

[18] Gandy, R. O., *An early proof of normalization by A.M. Turing*, in: J. Seldin and J. Hindley, editors, *To H.B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, pages 453–455, Academic Press (1980).

[19] Gandy, R. O., *Proofs of strong normalization*, in: J. Seldin and J. Hindley, editors, *To H.B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, pages 457–477, Academic Press (1980).

[20] Gardner, P., *Discovering needed reductions using type theory*, in: M. Hagiya and J. C. Mitchell, editors, *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).
https://doi.org/10.1007/3-540-57887-0_115

[21] Kfoury, A. J., *A linearization of the lambda-calculus and consequences*, J. Log. Comput. **10**, pages 411–436 (2000).
https://doi.org/10.1093/LOGCOM/10.3.411

[22] Kfoury, A. J. and J. B. Wells, *New notions of reduction and non-semantic proofs of beta-strong normalization in typed lambda-calculi*, in: *Proceedings, 10th Annual IEEE Symposium on Logic in Computer Science, San Diego, California, USA, June 26-29, 1995*, pages 311–321, IEEE Computer Society (1995).
https://doi.org/10.1109/LICS.1995.523266

[23] Klop, J. W., *Combinatory Reduction Systems*, Ph.D. thesis, Utrecht University (1980).
https://eprints.illc.uva.nl/id/eprint/1876/

[24] Krivine, J., *Lambda-calculus, types and models*, Ellis Horwood series in computers and their applications, Masson (1993), ISBN 978-0-13-062407-9.

[25] Liquori, L. and S. Ronchi Della Rocca, *Intersection-types à la church*, Inf. Comput. **205**, pages 1371–1386 (2007).
https://doi.org/10.1016/J.IC.2007.03.005

[26] Nederpelt, R. P., *Strong normalization in a typed lambda calculus with lambda structured types*, Phd thesis, TU Eindhoven (1973).
https://doi.org/10.6100/IR145802

[27] Neergaard, P. M. and M. H. Sørensen, *Conservation and uniform normalization in lambda calculi with erasing reductions*, Inf. Comput. **178**, pages 149–179 (2002).
https://doi.org/10.1006/inco.2002.3153

[28] Paolini, L., M. Piccolo and S. Ronchi Della Rocca, *Essential and relational models*, Math. Struct. Comput. Sci. **27**, pages 626–650 (2017).
https://doi.org/10.1017/S0960129515000316

[29] Pottinger, G., *A type assignment for the strong normalizable lambda-terms.*, in: J. Hindley and J. Seldin, editors, *To H.B. Curry: Essays on combinatory logic, lambda calculus and formalisms*, pages 561–577, Academic Press (1980).
https://doi.org/10.1109/LICS.1995.523266

[30] Ronchi Della Rocca, S. and L. Paolini, *The Parametric Lambda Calculus - A Metamodel for Computation*, Texts in Theoretical Computer Science. An EATCS Series, Springer (2004), ISBN 978-3-642-05746-5.
https://doi.org/10.1007/978-3-662-10394-4

[31] van Bakel, S., *Complete restrictions of the intersection type discipline*, Theor. Comput. Sci. **102**, pages 135–163 (1992).
https://doi.org/10.1016/0304-3975(92)90297-S

A Technical appendix

A.1 Proofs of Section 2 — An intrinsically typed presentation of idempotent intersection types

In this section we give detailed proofs of the results about the $\Lambda^{\mathtt{i}}_{\cap}$-calculus stated in Section 2.

A.1.1 Subject reduction

Proposition A.1 (Subject reduction) *If $t \to_{\mathtt{i}} t'$ and $\Gamma \vdash_{\mathtt{i}} t : A$ then $\Gamma \vdash_{\mathtt{i}} t' : A$.*

Proof. We generalize the statement, proving also that if $\vec{t} \to_{\mathtt{i}} \vec{t'}$ and $\Gamma \Vdash_{\mathtt{i}} \vec{t} : \vec{A}$ then $\Gamma \Vdash_{\mathtt{i}} \vec{t'} : \vec{A}$. The proof proceeds by induction on the definition of $\to_{\mathtt{i}}$.

1. If t is a redex, i.e, $t = (\lambda x^{\vec{B}}.\, s)\, \vec{u}$, then the derivation proving $\Gamma \vdash_{\mathtt{i}} t : A$ is:

$$\dfrac{\dfrac{\Gamma, x : \vec{B} \vdash_{\mathtt{i}} s : A}{\Gamma \vdash_{\mathtt{i}} \lambda x^{\vec{B}}.\, s : \vec{B} \to A} \quad \dfrac{}{\Gamma \Vdash_{\mathtt{i}} \vec{u} : \vec{B}}}{\Gamma \vdash_{\mathtt{i}} (\lambda x^{\vec{B}}.\, s)\, \vec{u} : A}$$

   By Substitution Lemma 2.16, $\Gamma, x : \vec{B} \vdash_{\mathtt{i}} s : A$ implies $\Gamma \vdash_{\mathtt{i}} s[x^{\vec{B}} := \vec{u}] : A$.
   Otherwise, it can be $t = \lambda x^{A}.\, s$, where $s \to_{\mathtt{i}} s'$, or $t = s\, \vec{u}$, where either $s \to_{\mathtt{i}} s'$ or $\vec{u} \to_{\mathtt{i}} \vec{u'}$; in all these cases the proof follows by IH.
2. $\vec{t} \to_{\mathtt{i}} \vec{t'}$ means $\vec{t} = \{t_1, .., t_n\}$, and $\vec{t'} = \{t'_1, ..., t'_n\}$, and there is i such that $t_i \to_{\mathtt{i}} t'_i$, and $t'_j = t_j$, for $j \neq i$. Then the proof follows by IH.

□

A.1.2 Confluence

Confluence will be proved by the methodology of parallel reduction.

Definition A.2 *1. The parallel reduction $\Rightarrow_{\mathtt{i}}$ is defined on terms and set-terms by the following rules, closed under contexts.*

$$\begin{array}{lll}
x^A & \Rightarrow_{\mathtt{i}} x^A & \\
\lambda x^{\vec{A}}.\, t & \Rightarrow_{\mathtt{i}} \lambda x^{\vec{A}}.\, t' & \texttt{if} \quad t \Rightarrow_{\mathtt{i}} t' \\
t\vec{s} & \Rightarrow_{\mathtt{i}} t'\vec{s'} & \texttt{if} \quad t \Rightarrow_{\mathtt{i}} t' \quad \texttt{and} \quad \vec{s} \Rightarrow_{\mathtt{i}} \vec{s'} \\
(\lambda x^{\vec{A}}.\, t)\vec{s} & \Rightarrow_{\mathtt{i}} t'[x^{\vec{A}} := \vec{s'}] & \texttt{if} \quad t \Rightarrow_{\mathtt{i}} t' \quad \texttt{and} \quad s \Rightarrow_{\mathtt{i}} \vec{s'} \\
\{t_1, \ldots, t_n\} & \Rightarrow_{\mathtt{i}} \{t'_1, \ldots, t'_n\} & \texttt{if} \quad t_i \Rightarrow_{\mathtt{i}} t'_i (1 \leq i \leq n)
\end{array}$$

Note that $\Rightarrow_{\mathtt{i}}$ is non-deterministic, since, when the term is a $\to_{\mathtt{i}}$ redex, either the third or the fourth rule can be applied. Roughly speaking, $\Rightarrow_{\mathtt{i}}$ corresponds to reduce simultaneously a subset of the visible redexes by the $\to_{\mathtt{i}}$ reduction.

2. The complete development of a term t, is the following function, which denotes the result of the simultaneous reduction of all visible redexes in a term.

$$\begin{array}{rl}
\mathtt{C}(x^A) & := x^A \\
\mathtt{C}(\lambda x^{\vec{A}}.\, t) & := \lambda x^{\vec{A}}.\, \mathtt{C}(t) \\
\mathtt{C}((\lambda x^{\vec{A}}.\, t)\, \vec{s}) & := \mathtt{C}(t)[x^{\vec{A}} := \mathtt{C}(\vec{s})] \\
\mathtt{C}(t\, \vec{s}) & := \mathtt{C}(t)\, \mathtt{C}(\vec{s}) \quad \textit{if } t \textit{ is not an abstraction} \\
\mathtt{C}(\{t_1, \ldots, t_n\}) & := \{\mathtt{C}(t_1), \ldots, \mathtt{C}(t_n)\}
\end{array}$$

The $\Rightarrow_{\mathtt{i}}$ reduction enjoys the following properties, whose proof is easy.

Proposition A.3 *1.* $t \to_{\mathtt{i}} t'$ *implies* $t \Rightarrow_{\mathtt{i}} t'$.

2. $t \Rightarrow_{\mathtt{i}} t'$ *implies* $t \to_{\mathtt{i}}^* t'$.

3. $\to_{\mathtt{i}}^*$ *is the transitive closure of* $\Rightarrow_{\mathtt{i}}$.

Lemma A.4 (Substitution) *Let* $t \Rightarrow_{\mathtt{i}} t'$ *and* $\vec{s} \Rightarrow_{\mathtt{i}} \vec{s}'$. *Then* $t[x^{\vec{A}} := \vec{s}] \Rightarrow_{\mathtt{i}} t'[x^{\vec{A}} := \vec{s}']$.

Proof. By induction on t. □

Lemma A.5 $t \Rightarrow_{\mathtt{i}} t'$ *implies* $t' \Rightarrow_{\mathtt{i}} \mathtt{C}(t)$.

Proof. By induction on t. Let $t = (\lambda x^{\vec{A}}.u)\vec{s}$. So t' is either $(\lambda x^{\vec{A}}.u')\vec{s}'$ or $u'[x^{\vec{A}} := \vec{s}']$, where $\vec{s} \Rightarrow_{\mathtt{i}} \vec{s}'$ and $u \Rightarrow_{\mathtt{i}} u'$. By IH, $u' \Rightarrow_{\mathtt{i}} \mathtt{C}(u)$ and $\vec{s}' \Rightarrow_{\mathtt{i}} \mathtt{C}(\vec{s})$. In both cases, $t' \to_{\mathtt{i}} \mathtt{C}(u)[x^{\vec{A}} := \mathtt{C}(\vec{s})] = \mathtt{C}(t)$, in the former case by IH, in the latter by IH and Lemma A.4. The other cases come directly from IH. □

The $\Rightarrow_{\mathtt{i}}$ reduction enjoys the diamond property, i.e.,

Lemma A.6 *If* $t \Rightarrow_{\mathtt{i}} t_1$ *and* $t \Rightarrow_{\mathtt{i}} t_2$, *then there is* t_3 *such that* $t_1 \Rightarrow_{\mathtt{i}} t_3$ *and* $t_2 \Rightarrow_{\mathtt{i}} t_3$.

Proof. By Lemma A.5, $t_3 = \mathtt{C}(t)$. □

Proposition A.7 (Confluence) *If* $t \to_{\mathtt{i}}^* s$ *and* $t \to_{\mathtt{i}}^* u$ *then there is* r *such that* $s \to_{\mathtt{i}}^* r$ *and* $u \to_{\mathtt{i}}^* r$.

Proof. By Proposition A.3.3 and Lemma A.6. □

A.2 Proofs of Section 3 — Strong normalization via a decreasing measure

In this section we give detailed proofs of the results about the $\Lambda^{\mathtt{im}}_{\cap}$-calculus stated in Section 3.

A.2.1 Subject reduction

Lemma A.8 (Substitution) *Let* $\Gamma, x : \vec{A} \vdash_{\mathtt{im}} t : B$ *and* $\Gamma \Vdash_{\mathtt{im}} \vec{s} : \vec{A}$. *Then* $\Gamma \vdash_{\mathtt{im}} t[x^{\vec{A}} := \vec{s}] : B$.

Proof. *By induction on* t, *generalizing the statement to set-terms. We consider only the wrapper case, since the remaining cases are exactly like those in Lemma 2.16. Let* $t = t_1\langle \vec{t_2} \rangle$. *Then* $\Gamma, x : \vec{A} \vdash_{\mathtt{im}} t_1\langle \vec{t_2} \rangle : B$ *concludes with* `im-wrap`, *so by inversion we have that* $\Gamma, x : \vec{A} \vdash_{\mathtt{im}} t_1 : B$ *and* $\Gamma, x : \vec{A} \Vdash_{\mathtt{im}} \vec{t_2} : \vec{C}$. *By IH,* $\Gamma \vdash_{\mathtt{im}} t_1[x^{\vec{A}} := \vec{s}] : B$ *and* $\Gamma \Vdash_{\mathtt{im}} \vec{t_2}[x^{\vec{A}} := \vec{s}] : \vec{C}$. *By* `im-wrap` *it follows that* $\Gamma \vdash_{\mathtt{im}} t_1[x^{\vec{A}} := \vec{s}]\langle \vec{t_2}[x^{\vec{A}} := \vec{s}] \rangle : B$. *By definition of substitution* $(t_1\langle \vec{t_2} \rangle)[x^{\vec{A}} := \vec{s}] = t_1[x^{\vec{A}} := \vec{s}]\langle \vec{t_2}[x^{\vec{A}} := \vec{s}] \rangle$, *hence we conclude that* $\Gamma \vdash_{\mathtt{im}} (t_1\langle \vec{t_2} \rangle)[x^{\vec{A}} := \vec{s}] : B$. □

Proposition A.9 (Subject reduction) *If* $t \to_{\mathtt{im}} t'$ *and* $\Gamma \vdash_{\mathtt{im}} t : A$, *then* $\Gamma \vdash_{\mathtt{im}} t' : A$.

Proof. Simultaneously by induction on the reduction relation $\to_{\mathtt{im}}$.

1. In the case $t = (\lambda x^{\vec{B}}.\, s)\mathtt{L}\,\vec{u} \to_{\mathtt{im}} s[x^{\vec{B}} := \vec{u}]\langle \vec{u} \rangle \mathtt{L} = t'$ we have that t must be typed with the following derivation scheme, where n is the amount of wrappers in $\mathtt{L}$:

$$\dfrac{\dfrac{\dfrac{\dfrac{\Gamma, x : \vec{B} \vdash_{\mathtt{im}} s : C}{\Gamma \vdash_{\mathtt{im}} \lambda x^{\vec{B}}.\, s : \vec{B} \to C}\,\mathtt{im\text{-}I}\to \qquad \Pi_1}{\mathtt{im\text{-}wrap}} \qquad \vdots \qquad \Pi_n}{\mathtt{im\text{-}wrap}} \qquad \Gamma \vdash_{\mathtt{im}} \vec{u} : \vec{B}}{\Gamma \vdash_{\mathtt{im}} (\lambda x^{\vec{B}}.\, s)\mathtt{L}\,\vec{u} : C}\,\mathtt{im\text{-}E}\to$$

   Hence from Lemma A.8, since $\Gamma, x : \vec{B} \vdash_{\mathtt{im}} s : C$ and $\Gamma \vdash_{\mathtt{im}} \vec{u} : \vec{B}$, we obtain $\Gamma \vdash_{\mathtt{im}} s[x^{\vec{B}} := \vec{u}] : C$. Then from the corresponding $n + 1$ applications of the `im-wrap` we obtain $\Gamma \vdash_{\mathtt{im}} s[x^{\vec{B}} := \vec{u}]\langle \vec{u} \rangle \mathtt{L} : C$.

2. If $t = \lambda x^{\vec{B}}.\, s \to_{\mathtt{im}} \lambda x^{\vec{B}}.\, s' = t'$ with $s \to_{\mathtt{im}} s'$, then $A = \vec{B} \to C$ and $\Gamma \vdash_{\mathtt{im}} \lambda x^{\vec{B}}.\, s : \vec{B} \to C$. By inversion of $\mathtt{im\text{-}I}\to$ it follows that $\Gamma, x : \vec{B} \vdash_{\mathtt{im}} s : C$. Then by IH we have that $\Gamma, x : \vec{B} \vdash_{\mathtt{im}} s' : C$, and by $\mathtt{im\text{-}I}\to$ we conclude $\Gamma \vdash_{\mathtt{im}} \lambda x^{\vec{B}}.\, s' : \vec{B} \to C$.
3. If $t = s\,\vec{u} \to_{\mathtt{im}} s'\,\vec{u} = t'$ with $s \to_{\mathtt{im}} s'$, then we have that $\Gamma \vdash_{\mathtt{im}} s\,\vec{u} : A$ and, by inversion of $\mathtt{im\text{-}E}\to$, it follows that $\Gamma \vdash_{\mathtt{im}} s : \vec{B} \to A$ and $\Gamma \Vdash_{\mathtt{im}} \vec{u} : \vec{B}$. Then by IH we have that $\Gamma \vdash_{\mathtt{im}} s' : \vec{B} \to A$, and by $\mathtt{im\text{-}E}\to$ we conclude $\Gamma \vdash_{\mathtt{im}} s'\,\vec{u} : A$.
4. If $t = s\,\vec{u} \to_{\mathtt{im}} s\,\vec{u}' = t'$ with $\vec{u} \to_{\mathtt{im}} \vec{u}'$, then we have that $\Gamma \vdash_{\mathtt{im}} s\,\vec{u} : A$ and, by inversion of $\mathtt{im\text{-}E}\to$, it follows that $\Gamma \vdash_{\mathtt{im}} s : \vec{B} \to A$ and $\Gamma \Vdash_{\mathtt{im}} \vec{u} : \vec{B}$. Then by IH we have that $\Gamma \Vdash_{\mathtt{im}} \vec{u}' : \vec{B}$, and by $\mathtt{im\text{-}E}\to$ we conclude $\Gamma \vdash_{\mathtt{im}} s\,\vec{u}' : A$.
5. If $\vec{t} = \{t_1, \ldots, t_i, \ldots, t_n\} \to_{\mathtt{im}} \{t_1, \ldots, t'_i, \ldots, t_n\} = \vec{t}'$ with $i \in 1..n$, $n \geq 1$ and $t_i \to_{\mathtt{im}} t'_i$, then we have that $\Gamma \Vdash_{\mathtt{im}} \{t_1, \ldots, t_i, \ldots, t_n\} : \{A_1, \ldots, A_i, \ldots A_n\}$. By inversion of rule $\mathtt{im\text{-}many}$, it follows that $\Gamma \vdash_{\mathtt{im}} t_1 : A_1, \ldots, \Gamma \vdash_{\mathtt{im}} t_i : A_i, \ldots, \Gamma \vdash_{\mathtt{im}} t_n : A_n$. Then by IH we have that $\Gamma \vdash_{\mathtt{im}} t'_i : A_i$, and by $\mathtt{im\text{-}many}$ we conclude $\Gamma \Vdash_{\mathtt{im}} \{t_1, \ldots, t'_i, \ldots, t_n\} : \{A_1, \ldots, A_i, \ldots A_n\}$.
6. If $t = s\langle\vec{u}\rangle \to_{\mathtt{im}} s'\langle\vec{u}\rangle = t'$ with $s \to_{\mathtt{im}} s'$, then we have that $\Gamma \vdash_{\mathtt{im}} s\langle\vec{u}\rangle : A$ and, by inversion of $\mathtt{im\text{-}wrap}$, it follows that $\Gamma \vdash_{\mathtt{im}} s : A$ and $\Gamma \Vdash_{\mathtt{im}} \vec{u} : \vec{B}$. Then by IH we have that $\Gamma \vdash_{\mathtt{im}} s' : A$, and by $\mathtt{im\text{-}wrap}$ we conclude $\Gamma \vdash_{\mathtt{im}} s'\langle\vec{u}\rangle : A$.
7. If $t = s\langle\vec{u}\rangle \to_{\mathtt{im}} s\langle\vec{u}'\rangle = t'$ with $\vec{u} \to_{\mathtt{im}} \vec{u}'$, then we have that $\Gamma \vdash_{\mathtt{im}} s\langle\vec{u}\rangle : A$ and, by inversion of $\mathtt{im\text{-}wrap}$, it follows that $\Gamma \vdash_{\mathtt{im}} s : A$ and $\Gamma \Vdash_{\mathtt{im}} \vec{u} : \vec{B}$. Then by IH we have that $\Gamma \Vdash_{\mathtt{im}} \vec{u}' : \vec{B}$, and by $\mathtt{im\text{-}wrap}$ we conclude $\Gamma \vdash_{\mathtt{im}} s\langle\vec{u}'\rangle : A$.

□

A.2.2 Confluence

We extend the proof of Proposition 2.19 by adding the cases involving wrappers and modifying those involving $\mathtt{i}-$redexes by $\mathtt{im}-$redexes (*i.e.* allowing a list of wrappers $\mathtt{L}$ between the w-abstraction and the argument).

Definition A.10

1. *The parallel reduction $\Rightarrow_{\mathtt{im}}$ is extended by modifying the β case to allow lists of wrappers, and by adding the congruence cases of wrappers and lists of wrappers:*

$$\begin{array}{lll}
(\lambda x^{\vec{A}}.\, t)\mathtt{L}\vec{s} & \Rightarrow_{\mathtt{im}} t'[x^{\vec{A}} := \vec{s}']\langle\vec{s}'\rangle\mathtt{L}' & \text{if } t \Rightarrow_{\mathtt{im}} t',\ s \Rightarrow_{\mathtt{im}} \vec{s}' \text{ and } \mathtt{L} \Rightarrow_{\mathtt{im}} \mathtt{L}' \\
t\langle\vec{s}\rangle & \Rightarrow_{\mathtt{im}} t'\langle\vec{s}'\rangle & \text{if } t \Rightarrow_{\mathtt{im}} t' \text{ and } \vec{s} \Rightarrow_{\mathtt{im}} \vec{s}' \\
\Box\langle\vec{t}_1\rangle \ldots \langle\vec{t}_n\rangle & \Rightarrow_{\mathtt{im}} \Box\langle\vec{t}'_1\rangle \ldots \langle\vec{t}'_n\rangle & \text{if } \vec{t}_i \Rightarrow_{\mathtt{im}} \vec{t}'_i \text{ for all } i \in 1..n
\end{array}$$

2. *The complete development of a term t is extended by modifying the β case to allow lists of wrappers, and by adding the congruence cases of wrappers and lists of wrappers*

$$\begin{array}{rcl}
\mathtt{C}((\lambda x^{\vec{A}}.\, t)\mathtt{L}\vec{s}) & := & \mathtt{C}(t)[x^{\vec{A}} := \mathtt{C}(\vec{s})]\langle\mathtt{C}(\vec{s})\rangle\mathtt{C}(\mathtt{L}) \\
\mathtt{C}(t\langle\vec{s}\rangle) & := & \mathtt{C}(t)\langle\mathtt{C}(\vec{s})\rangle \\
\mathtt{C}(\Box\langle\vec{t}_1\rangle \ldots \langle\vec{t}_n\rangle) & := & \Box\langle\mathtt{C}(t_1)\rangle, \ldots, \langle\mathtt{C}(t_n)\rangle
\end{array}$$

As well as $\Rightarrow_{\mathtt{i}}$ reduction, $\Rightarrow_{\mathtt{im}}$ reduction enjoys the following properties:

Proposition A.11 (Properties of $\Rightarrow_{\mathtt{im}}$)

1. *$t \to_{\mathtt{im}} t'$ implies $t \Rightarrow_{\mathtt{im}} t'$.*
2. *$t \Rightarrow_{\mathtt{im}} t'$ implies $t \to^*_{\mathtt{im}} t'$.*
3. *$\to^*_{\mathtt{im}}$ is the transitive closure of $\Rightarrow_{\mathtt{im}}$.*

Lemma A.12 (Substitution) $t[y^{\vec{A}} := \vec{s}][x^{\vec{B}} := \vec{u}] = t[x^{\vec{B}} := \vec{u}][y^{\vec{A}} := \vec{s}[x^{\vec{B}} := \vec{u}]]$ *if* $y \notin \mathsf{fv}(\vec{u})$

Proof. By induction on t. □

Lemma A.13 (Compatibility of substitution and parallel reduction) *Let* $t \Rightarrow_{\mathtt{im}} t'$ *and* $\vec{s} \Rightarrow_{\mathtt{im}} \vec{s'}$. *Then* $t[x^{\vec{A}} := \vec{s}] \Rightarrow_{\mathtt{im}} t'[x^{\vec{A}} := \vec{s'}]$.

Proof. By induction on t, where the interesting case is that of $t = (\lambda y^{\vec{A}}.\, u)\mathtt{L}\vec{r}$. Then t' depends on whether the head $\mathtt{im}$−redex is contracted or not. In both cases it holds that $u \Rightarrow_{\mathtt{im}} u'$, $\vec{r} \Rightarrow_{\mathtt{im}} \vec{r'}$, and $\mathtt{L} \Rightarrow_{\mathtt{im}} \mathtt{L}'$.

1. If $t' = (\lambda y^{\vec{B}}.\, u')\mathtt{L}'\vec{r'}$, then, by definition of substitution and IH, we have that

$$\begin{aligned}((\lambda y^{\vec{B}}.\, u)\mathtt{L}\vec{r})[x^{\vec{A}} := \vec{s}] \;=\;& (\lambda y^{\vec{B}}.\, u[x^{\vec{A}} := \vec{s}])\mathtt{L}[x^{\vec{A}} := \vec{s}]\vec{r}[x^{\vec{A}} := \vec{s}] \\ \Rightarrow_{\mathtt{im}}\;& (\lambda y^{\vec{B}}.\, u'[x^{\vec{A}} := \vec{s'}])\mathtt{L}'[x^{\vec{A}} := \vec{s'}]\vec{r'}[x^{\vec{A}} := \vec{s'}] \\ =\;& ((\lambda y^{\vec{B}}.\, u')\mathtt{L}'\vec{r'})[x^{\vec{A}} := \vec{s'}]\end{aligned}$$

2. If $t' = u'[y^{\vec{B}} := \vec{r'}]\langle\vec{r'}\rangle\mathtt{L}'$, then, by definition of substitution, IH, and Lemma A.12, we have that

$$\begin{aligned}((\lambda y^{\vec{B}}.\, u)\mathtt{L}\vec{r})[x^{\vec{A}} := \vec{s}] \;=\;& (\lambda y^{\vec{B}}.\, u[x^{\vec{A}} := \vec{s}])\mathtt{L}[x^{\vec{A}} := \vec{s}]\vec{r}[x^{\vec{A}} := \vec{s}] \\ \Rightarrow_{\mathtt{im}}\;& u'[x^{\vec{A}} := \vec{s'}][y^{\vec{B}} := \vec{r'}[x^{\vec{A}} := \vec{s'}]]\langle\vec{r'}[x^{\vec{A}} := \vec{s'}]\rangle\mathtt{L}'[x^{\vec{A}} := \vec{s'}] \\ =\;& u'[y^{\vec{B}} := \vec{r'}][x^{\vec{A}} := \vec{s'}]\langle\vec{r'}[x^{\vec{A}} := \vec{s'}]\rangle\mathtt{L}'[x^{\vec{A}} := \vec{s'}] \\ =\;& (u'[y^{\vec{B}} := \vec{r'}]\langle\vec{r'}\rangle\mathtt{L}')[x^{\vec{A}} := \vec{s}]\end{aligned}$$

□

Lemma A.14 (Composition lemma for wrappers) *If* $t \Rightarrow_{\mathtt{im}} t'$ *and* $\mathtt{L} \Rightarrow_{\mathtt{im}} \mathtt{L}'$, *then* $t\mathtt{L} \Rightarrow_{\mathtt{im}} t'\mathtt{L}'$.

Proof. Let $\mathtt{L} = \Box\langle\vec{t_1}\rangle \dots \langle\vec{t_n}\rangle$. We proceed by induction on n.

1. If $n = 0$, then $t\mathtt{L}t\Box = t \Rightarrow_{\mathtt{im}} t' = t'\Box = t'\mathtt{L}'$.
2. If $n = k + 1$, then by IH we have that $t\Box\langle t_1\rangle \dots \langle t_k\rangle \Rightarrow_{\mathtt{im}} t'\Box\langle t'_1\rangle \dots \langle t'_k\rangle$, hence

$$\begin{aligned}t\mathtt{L} \;=\;& t\langle t_1\rangle \dots \langle t_k\rangle\langle t_{k+1}\rangle \\ =\;& t\langle t_1\rangle \dots \langle t_k\rangle\langle t_{k+1}\rangle \\ \Rightarrow_{\mathtt{im}}\;& t'\langle t'_1\rangle \dots \langle t'_k\rangle\langle t'_{k+1}\rangle \\ =\;& t'\Box\langle t'_1\rangle \dots \langle t'_k\rangle\langle t'_{k+1}\rangle \\ =\;& t'\mathtt{L}'\end{aligned}$$

□

Lemma A.15 $t \Rightarrow_{\mathtt{im}} t'$ *implies* $t' \Rightarrow_{\mathtt{im}} \mathtt{C}(t)$.

Proof. By induction on t. The interesting case is that of $t = (\lambda x^{\vec{A}}.u)\mathtt{L}\vec{s}$. Then t' depends on whether the head $\mathtt{im}$-redex is contracted or not. In both cases it holds that $u \Rightarrow_{\mathtt{im}} u'$, $\vec{s} \Rightarrow_{\mathtt{im}} \vec{s'}$, and $\mathtt{L} \Rightarrow_{\mathtt{im}} \mathtt{L}'$.

1. If $t' = (\lambda x^{\vec{A}}.u')\mathtt{L}'\vec{s'}$, then by IH we have that

$$\begin{aligned}(\lambda x^{\vec{A}}.u)\mathtt{L}\vec{s} \Rightarrow_{\mathtt{im}}\;& (\lambda x^{\vec{A}}.u')\mathtt{L}'\vec{s'} \\ \Rightarrow_{\mathtt{im}}\;& (\lambda x^{\vec{A}}.\mathtt{C}(u))\mathtt{C}(\mathtt{L})\mathtt{C}(\vec{s}) \\ \to_{\mathtt{im}}\;& \mathtt{C}(u)[x^{\vec{A}} := \mathtt{C}(\vec{s})]\langle\mathtt{C}(\vec{s})\rangle\mathtt{C}(\mathtt{L}) \\ =\;& \mathtt{C}((\lambda x^{\vec{A}}.u)\mathtt{L}\vec{s})\end{aligned}$$

2. If $t' = u'[x^{\vec{A}} := \vec{s'}]\langle\vec{s'}\rangle\mathtt{L}'$, then by IH and Lemma A.13 we have that

$$\begin{aligned}(\lambda x^{\vec{A}}.u)\mathtt{L}\vec{s} &\Rightarrow_{\mathtt{im}} u'[x^{\vec{A}} := \vec{s'}]\langle\vec{s'}\rangle\mathtt{L}' \\ &\Rightarrow_{\mathtt{im}} \mathtt{C}(u)[x^{\vec{A}} := \mathtt{C}(\vec{s})]\langle\mathtt{C}(\vec{s})\rangle\mathtt{C}(\mathtt{L}) \\ &= \quad \mathtt{C}((\lambda x^{\vec{A}}.u)\mathtt{L}\vec{s})\end{aligned}$$

The remaining cases come directly from IH. □

The $\Rightarrow_{\mathtt{im}}$ reduction enjoys the diamond property, i.e.,

Lemma A.16 *If $t \Rightarrow_{\mathtt{im}} t_1$ and $t \Rightarrow_{\mathtt{im}} t_2$, then there is t_3 such that $t_1 \Rightarrow_{\mathtt{im}} t_3$ and $t_2 \Rightarrow_{\mathtt{im}} t_3$.*

Proof. By Lemma A.15, $t_3 = \mathtt{C}(t)$. □

Proposition A.17 (Confluence) *If $t_1 \to^*_{\mathtt{im}} t_2$ and $t_1 \to^*_{\mathtt{im}} t_3$, there exists a term t_4 such that $t_2 \to^*_{\mathtt{im}} t_4$ and $t_3 \to^*_{\mathtt{im}} t_4$.*

Proof. By Lemmas A.11.3 and A.16. □