A big-step operational semantics via non-idempotent intersection types

Alexis Bernadet[1] and Stéphane Graham-Lengrand[2]

1,2 École Polytechnique, France
2 CNRS, France

Abstract

We present a typing system of non-idempotent intersection types that characterises strongly normalising λ-terms and can been seen as a big-step operational semantics: we prove that a strongly normalising λ-term accepts, as its type, the structure of its normal form. As a by-product of identifying such semantical components in typing trees, we are able to define a trivial measure (the number of times a typing rule is applied) that exactly captures the length of the longest β-reduction sequence starting from a given typable term.

1 Introduction

Operational semantics describes the behaviour of programs using syntax, in other words without (explicitly) constructing a denotational model (for example, a Scott Domain [15]). While small-steps operational semantics is based on the step-by-step evaluation of a program (usually by means of a rewrite system or an abstract machine), a big-step operational semantics directly relates a program to its final value / normal form (if it exists): judgements of the form $M \downarrow v$ (where M is a program and v the value of M after computation) can be derived with inference rules such as

$$\frac{M_1 \downarrow \lambda x.M_3 \quad M_2 \downarrow M_4 \quad M_3\{x := \ M_4\} \downarrow M_5}{M_1 M_2 \downarrow M_5}$$

in the case of λ-calculus.

In this paper we provide a typing system for the λ-calculus that can serve as an inference system deriving a big-step semantics for all strongly normalising λ-terms.

Not only is this system following the traditional concepts and notations of typing, it more fundamentally differs from standard big-step semantics in that at no point of derivation trees is a substitution ever computed (unlike the rule above).

Moreover, while most big-step semantics correspond to an (often strict) evaluation strategy that is neither the least nor the most efficient, our typing system is related to longest β-reduction sequences. It therefore provides a semantics to strongly normalising terms only, but on the other hand it provides quantitative information about reduction: a trivial measure on the typing derivation relating a term M to its semantics provides the exact length of the longest β-reduction sequence starting from M.

To build this semantics for all strongly normalising terms, we naturally use intersection types. Originally introduced in [5], intersection types extend simple types with the constructor $\cap$ (if A and B are types, then $A \cap B$ is a type) and additional typing rules such as:

$$\frac{M : A \quad M : B}{M : A \cap B}$$

Intersection type systems relate typing to various evaluation properties of λ-terms (see e.g. [2, 12, 16, 10, 8, 9, 1]). Most importantly for us, they can be tuned to characterise strongly normalising terms (a term is typable if and only if it is strongly normalising).

Moreover, we can consider types as information about how a λ-term is used and $M : A$ means that M will be used as A. Then, $M : A \cap B$ means that M will be used as A and B. One way or another, typing systems using intersections usually feature *idempotency*: $A \cap A$ is equivalent to A. If we choose to drop idempotency, then we can distinguish $M : A$ and $M : A \cap A$ (M will be used as A exactly twice), and a typing judgement gives us quantitative information. This idea is mainly inspired by linear logic [11] and denotational models built on multisets.

For example, de Carvalho [6, 7] introduced a typing system with non-idempotent intersections, and found a relation between the size of a typing tree and the length of linear head-reduction sequences (as can be found in some abstract machines).

In a previous paper [3], we gave a typing system that (unlike de Carvalho's) characterises strongly normalising terms, and a trivial measure on typing trees (the number of uses of the rule typing applications) strictly decreases with each β-reduction. Therefore, strong normalisation is a corollary of Subject Reduction and the measure provides a bound on the length of the longest β-reduction sequence.

Refining the bound into an exact measurement was the challenge of the latter part of [3], which provided a convoluted solution: First, the exact length of the longest reduction sequence could only be obtained by considering the typing trees satisfying a particular condition called *optimality*. Then, another quantitative information about typing trees, the *degree*, had to be read, and the exact length was finally computed as *the measure of an optimal typing tree of smallest degree minus that degree.* Moreover, the result was obtained indirectly via the study of the above notions in Church-Klop's λI-calculus and then transposed into the pure λ-calculus by a sophisticated simulation.

However, it can be noticed that

- the degree is a quantitative data related to the normal form of a term;
- going via the λI-calculus is required because longest β-reduction sequences in the pure λ-calculus sometimes have to erase sub-terms in normal form.

The present paper develops the idea of incorporating information about normal forms into typing trees: for every normal form M, its structure v yields a special type $[v]$.[1]

Pushing this idea further leads to a non-idempotent intersection typing system where typing trees can be viewed as the derivations of a big-step semantics.

As a by-product, the system improves the results of [3] and simplifies their proofs: the length of the longest β-reduction sequence starting from a pure and strongly normalising λ-term M is simply the number of times a typing rule occurs in an optimal typing tree for M. The notion of optimality itself is actually much simpler, the notion of degree becomes useless, and no detour via λI is required.

The trick is that the typing tree for a given term directly indicates which λ-abstractions and applications will be consumed by β-reduction and which will remain in the normal form. These additional typing rules ensure that the measure of an optimal typing tree for a term in normal form is zero.

This enhancement of the typing system does not make the proofs of Soundness (typable implies strongly normalising) and Completeness (strongly normalising implies typable) more complicated.

In contrast to other semantics built with intersection types (for example [2, 4]), this semantics does not use filters of intersection types. However, since the typing system is different, we have to do the study of the typing system entirely, especially because it is a little less syntax-directed. On the other hand, this paper is self-contained: the reader does not need to have read [3] to understand definitions, lemmas, theorems and proofs.

In Section 2, we give the basic definitions and properties. In Section 3, we prove Soundness (typable implies strongly normalising) and Completeness (strongly normalising implies typable). In Section 4, we give more refined properties in the case where the typing tree is optimal. In particular, we prove the Complexity Result (Theorem 33) and the relation between the type of a term and its normal form (Theorem 35). In Section 5, we give a brief study of alternative definitions of the typing system. In Appendix A, we give detailed proofs of some lemmas and theorems.

[1] The structure of a normal form is the normal form where all variable names are forgotten.

$$\frac{}{(\lambda x.M)N \longrightarrow_\beta M\{x := N\}} \qquad \frac{M \longrightarrow_\beta M'}{\lambda x.M \longrightarrow_\beta \lambda x.M'}$$

$$\frac{M \longrightarrow_\beta M'}{MN \longrightarrow_\beta M'N} \qquad \frac{N \longrightarrow_\beta N'}{MN \longrightarrow_\beta MN'}$$

Figure 1 β-reduction

2 Basic definitions and properties

In Section 2.1, we review basic concepts of λ-calculus. In Section 2.2, we define the intersection types used in this paper, the contexts of the typing judgements and we prove some basic properties. In Section 2.3, we present the typing system and its basics properties.

2.1 Syntax

We review the λ-calculus and the concepts that are of interest for this paper.

▶ **Definition 1** (λ-terms). Terms are defined in the usual way with the following grammar:

$$M, N ::= x \mid \lambda x.M \mid MN$$

Free variables $\mathrm{fv}(M)$ and substitutions $M\{x := N\}$ are defined in the usual way and terms are considered up to α-equivalence.

The notion of reduction is the usual β-reduction:

▶ **Definition 2** (β-reduction). $M \longrightarrow_\beta M'$ is defined inductively by the rules of Figure 1.

To make some proofs and theorems about strong normalisation more readable, we use the following notations:

▶ **Definition 3** (Strongly normalising terms). Assume n is an integer.

We write **SN** for the set of strongly normalising terms for $\longrightarrow_\beta$.

We write $\mathbf{SN}_{=n}$ for the set of strongly normalising terms for $\longrightarrow_\beta$ such that the length of longest β-reduction sequence is equal to n.

The set $\mathbf{SN}_{\leq n}$ is defined by:

$$\mathbf{SN}_{\leq n} := \bigcup_{m \leq n} \mathbf{SN}_{=m}$$

We now proceed towards the notion of perpetual reduction which is used:

- to express the longest reduction sequence in Theorem 33
- as a reduction strategy to prove Theorem 29

In order to define it formally, we also define $\mathrm{acc}(M)$ (that characterize terms of the form x $M_1 \ldots M_n$) and $\Rrightarrow$ as follows:

▶ **Definition 4** (Accumulators and perpetual reduction). $\mathrm{acc}(M)$ is inductively defined by the following rules:

$$\frac{}{\mathrm{acc}(x)} \qquad \frac{\mathrm{acc}(M)}{\mathrm{acc}(MN)}$$

$M \Rrightarrow M'$ and $M \rightsquigarrow M'$ are mutually defined by the rules of Figure 2.

Notice that this is the same perpetual reduction strategy used in [3]. However, the presentation is different (we did not used $\Rrightarrow$). The presentation given in this paper avoids informal notations such as $xM_1 \ldots M_n$, which is here very cumbersome for case analyses: since there are two ways to type an application, there are 2^n ways to type the above.

Examples for $\rightsquigarrow$:

$$\frac{x \in \mathrm{fv}(M)}{(\lambda x.M)N \Rightarrow M\{x := N\}} \qquad \frac{x \notin \mathrm{fv}(M) \quad N \rightsquigarrow N'}{(\lambda x.M)N \Rightarrow (\lambda x.M)N'}$$

$$\frac{x \notin \mathrm{fv}(M) \quad N \text{ cannot be reduced by } \longrightarrow_\beta}{(\lambda x.M)N \Rightarrow M}$$

$$\frac{M \Rightarrow M'}{MN \Rightarrow M'N} \qquad \frac{\mathrm{acc}(M) \quad N \rightsquigarrow N'}{MN \Rightarrow MN'}$$

$$\frac{M \Rightarrow M'}{M \rightsquigarrow M'} \qquad \frac{M \rightsquigarrow M'}{\lambda x.M \rightsquigarrow \lambda x.M'}$$

Figure 2 Perpetual reduction

- $(\lambda x.M)NN_1 \dots N_n \rightsquigarrow M\{x := N\}N_1 \dots N_n$ if $x \in \mathrm{fv}(M)$.
- If $M \rightsquigarrow M'$, then $xM \rightsquigarrow xM'$.
- It is possible to have $M \rightsquigarrow M'$ without having $(\lambda x.M)N \rightsquigarrow (\lambda x.M')N$. For example, we have $(\lambda y.a)(xx) \rightsquigarrow a$ but we do not have $(\lambda x.(\lambda y.a)(xx))(\lambda z.zz) \rightsquigarrow (\lambda x.a)(\lambda z.zz)$. However, we do have $(\lambda x.(\lambda y.a)(xx))(\lambda z.zz) \rightsquigarrow (\lambda y.a)((\lambda z.zz)(\lambda z.zz))$.

Every lemma and theorem based on a syntactical analysis of a term in normal term uses the following lemma:

▶ **Lemma 5** (Shape of a normal form). *If M cannot be reduced by $\longrightarrow_\beta$, then*
- *either we have $acc(M)$,*
- *or M is of the form $\lambda x.M_1$.*

Proof. By induction on M. See Appendix A. ◀

One of the reasons why $\rightsquigarrow$ is used in Theorems 29 and 33, is that it is possible to reach the normal form by using $\rightsquigarrow$ only. More formally:

▶ **Lemma 6** (Applicability of $\rightsquigarrow$). *If M can be reduced by $\longrightarrow_\beta$, then M can be reduced by $\rightsquigarrow$.*

Proof. We prove by induction on M that if M can be reduced by $\longrightarrow_\beta$ then:
- If M is of the form $\lambda x.M_1$, then there exists M' such that $M \rightsquigarrow M'$.
- If not, then there exists M' such $M \Rightarrow M'$.

Therefore, in both cases there exists M' such that $M \rightsquigarrow M'$. See Appendix A. ◀

An optimal typing tree for a term M does not give the exact normal form M' of M but the *structure* of M' (Theorem 35), which is M' where the names of variables are forgotten. More formally:

▶ **Definition 7** (Structures). Structures v are defined by the following grammar:

$$\begin{array}{lcl} v & ::= & \lambda v \mid k \\ k & ::= & \triangledown \mid kv \end{array}$$

For every term M, struct(M) is partially defined as follows:

$$\begin{array}{lcll} \mathrm{struct}(x) & := & \triangledown & \\ \mathrm{struct}(\lambda x.M) & := & \lambda\mathrm{struct}(M) & \\ \mathrm{struct}(MN) & := & \mathrm{struct}(M)\mathrm{struct}(N) & (\mathrm{struct}(M) \text{ is of the form } k) \end{array}$$

struct(M) is coherent with α-equivalence because: If struct(M) is well-defined, then struct$(M\{x := y\})$ is well-defined and struct$(M\{x := y\}) = $ struct(M).

▶ **Lemma 8** (Structure of a normal term). *M cannot be reduced by $\longrightarrow_\beta$ if and only if struct(M) is well-defined. Moreover, if we have acc(M), then struct(M) is of the form k.*

Proof. By induction on M. See Appendix A. ◀

▶ Remark.
- $\mathrm{struct}(\lambda f\lambda x.f^n x) = \lambda\lambda\triangledown^n\triangledown$.
  Hence, Church's integers all have different structures.
- Structures cannot distinguish Church's booleans:
  $\mathrm{struct}(\lambda x.\lambda y.x) = \mathrm{struct}(\lambda x.\lambda y.y) = \lambda\lambda\triangledown$. However, this problem disappears if booleans are encoded as Church's integers (0 for "false" and 1 for "true"). More generally, the inhabitants of any enumerated type can be encoded in a way such that their structures remain distinct.
- Similarly, Church's encoding of pairs is preserved in the structures:
  Assume M and N cannot be reduced. We have $\mathrm{struct}(\lambda x.xMN) = \lambda\triangledown\mathrm{struct}(M)\mathrm{struct}(N)$. Therefore, if $\mathrm{struct}(\lambda x.xMN) = \mathrm{struct}(\lambda x.xM'N')$, then $\mathrm{struct}(M) = \mathrm{struct}(M')$ and $\mathrm{struct}(N) = \mathrm{struct}(N')$.
- Consequently, the inhabitants of any algebraic data type (for example lists, trees, etc ...) can be encoded in a way such that their structures remain disinct.
  Therefore structures can be considered a reasonable semantics.

2.2 Intersection types and contexts

First, we define the intersection types used in this paper and we show basic definitions and properties about them. Second, we define the contexts used in the typing derivations used in this paper and we show basic definitions and properties about them.

2.2.1 Intersection types

In this paper, intersection types are defined as follows:

▶ **Definition 9** (Intersection types).
F-types, A-types and U-types are defined by grammar on the right, where v ranges over structures (Definiton 7). With this grammar, $U \cap V$ is defined if and only if U and V are A-types. Therefore, by defining $A \cap \omega := A$, $\omega \cap A := A$ and $\omega \cap \omega := \omega$, we have $U \cap V$ defined for all U and V.

$$\begin{array}{lcl} F, G & ::= & [v] \mid A \to F \\ A, B, C & ::= & F \mid A \cap B \\ U, V & ::= & A \mid \omega \end{array}$$

Some remarks:
- The property that, in an arrow $A \to B$, B is not an intersection, is the standard restriction of *strict* types [16] and is used here to prove Lemma 20.1.
- ω is a device that allows synthetic formulations of definitions, properties and proofs: U-types are not defined by mutual induction with A-types and F-types, but separately, and we could have written the paper without them (only with more cases in statements and proofs).
- $[v]$ is a type used to give information about the normal form of a term and is used to type applications and abstractions that are not meant to be used in a β-reduction.

For example, $([\triangledown(\lambda\triangledown)] \to [\lambda\triangledown]) \cap [\triangledown]$ is an A-type.

To define the notion of *optimality*, which is used in Theorems 33 and 35, we need to define the notions of *input* types and *output* types:
- An input is an intersection of $[\triangledown]$.
- An output is of the form $[v]$.

More formally:

▶ **Definition 10** (Inputs and outputs).
The judgement input(U) is defined with the following rules:

$$\frac{}{\mathrm{input}([\triangledown])} \qquad \frac{\mathrm{input}(A) \quad \mathrm{input}(B)}{\mathrm{input}(A \cap B)} \qquad \frac{}{\mathrm{input}(\omega)}$$

We write output(F) if and only if F is of the form $[v]$.

$$\frac{}{F \approx F} \qquad \frac{}{A \cap B \approx B \cap A} \qquad \frac{}{(A \cap B) \cap C \approx A \cap (B \cap C)}$$

$$\frac{}{A \cap (B \cap C) \approx (A \cap B) \cap C} \qquad \frac{A \approx A' \quad B \approx B'}{A \cap B \approx A' \cap B'} \qquad \frac{}{\omega \approx \omega} \qquad \frac{A \approx B \quad B \approx C}{A \approx C}$$

Figure 3 Equivalence between types

To prove Subject Reduction and Subject Expansion (Theorems 23, 32 and 28) by using Lemmas 22 and 25, we have to define equivalence $\approx$ and inclusion $\subseteq$ between types. Here is the formal definitions and basic properties (notice that we do not have $A \approx A \cap A$):

▶ **Definition 11** (Equivalence between types).
Assume U and V are U-types. We define $U \approx V$ with the rules given in Figure 3.

The fact that $\approx$ is an equivalence relation can be easily proved (Lemma 12.4). Therefore, adding rules for reflexivity, symmetry and transitivity is superfluous and only adds more cases to treat in the proofs of statements where such a relation is assumed (e.g. Lemma 20.2).

We could have represented intersection types with multisets (and correspondingly used the type inclusion symbol $\subseteq$ the other way round). We chose to keep the standard notation $A \cap B$, with the corresponding inclusion satisfying $A \cap B \subseteq A$, since these are interpreted as set intersection and set inclusion in some models (e.g. realisability candidates). This way, we can also keep the equivalence relation explicit in the rest of the paper, which gives finer-grained results. For example, the equivalence only appears where it is necessary and the proof of Lemma 20.2 shows the mechanism that propagates the equivalence through the typing trees. These presentation choices are irrelevant to the key ideas of the paper.

▶ **Lemma 12** (Properties of $\approx$).
1. *Neutrality of ω: $U \cap \omega = \omega \cap U = U$.*
2. *Strictness of F-types: If $U \approx F$, then $U = F$.*
3. *Strictness of ω: If $U \approx \omega$, then $U = \omega$.*
4. *$\approx$ is an equivalence relation.*
5. *Commutativity of $\cap$: $U \cap V \approx V \cap U$*
6. *Associativity of $\cap$: $U_1 \cap (U_2 \cap U_3) \approx (U_1 \cap U_2) \cap U_3$*
7. *Stability of $\cap$: If $U \approx U'$ and $V \approx V'$, then $U \cap V \approx U' \cap V'$.*
8. *If $U \cap V = \omega$, then $U = V = \omega$.*
9. *If $U \cap V \approx U$, then $V = \omega$.*
10. *If $U \approx V$ and input(U), then input(V).*
11. *input$(U \cap V)$ if and only if input(U) and input(V).*

Proof. Similar to the proof in [4]. See Appendix A. ◀

▶ **Definition 13** (Sub-typing).
Assume U and V are U-types. We write $U \subseteq V$ if and only if there exists a U-type U' such that $U \approx V \cap U'$.

▶ **Lemma 14** (Properties of $\subseteq$).
1. *$\subseteq$ is a partial pre-order and $\approx$ is the equivalence relation associated to it: $U \subseteq V$ and $V \subseteq U$ if and only if $U \approx V$.*
2. *Projections: $U \cap V \subseteq U$ and $U \cap V \subseteq V$.*
3. *Stability of $\cap$: If $U \subseteq U'$ and $V \subseteq V'$, then $U \cap V \subseteq U' \cap V'$.*
4. *Greatest element: $U \subseteq \omega$.*
5. *If $U \subseteq V$ and input(U), then input(V).*

Proof. Straightforward. ◀

2.2.2 Contexts

To define typing judgements (Definition 18), we need to define contexts and give their basic properties. We naturally define pointwise the notion of equivalence, inclusion and input for contexts. More formally:

▶ **Definition 15** (Contexts).
A *context* Γ is a total map from the set of variables to the set of U-types such that the domain of Γ defined by:

$$\mathsf{Dom}(\Gamma) := \{x \mid \Gamma(x) \neq \omega\}$$

is finite.
$\cap$, $\approx$ and $\subseteq$ for contexts are defined pointwise:

$$\begin{array}{lcl} (\Gamma \cap \Delta)(x) & := & \Gamma(x) \cap \Delta(x) \\ \Gamma \approx \Delta & \Leftrightarrow & \forall x, \Gamma(x) \approx \Delta(x) \\ \Gamma \subseteq \Delta & \Leftrightarrow & \forall x, \Gamma(x) \subseteq \Delta(x) \end{array}$$

Notice that if $\mathsf{Dom}(\Gamma)$ and $\mathsf{Dom}(\Delta)$ are finite, then $\mathsf{Dom}(\Gamma \cap \Delta)$ is finite. Therefore $\Gamma \cap \Delta$ is indeed a context in the case where Γ and Δ are contexts.
The empty context $()$ is defined as follows: $()(x) := \omega$ for all x.
Assume Γ is a context, $x_1, \ldots, x_n$ are distinct variables and $U_1, \ldots U_n$ are U-types such that for all i, $x_i \notin \mathsf{Dom}(\Gamma)$. Then, the context $(\Gamma, x_1 : U_1, \ldots, x_n : U_n)$ is defined as follows:

$$\begin{array}{lcl} (\Gamma, x_1 : U_1, \ldots, x_n : U_n)(x_i) & := & U_i \\ (\Gamma, x_1 : U_1, \ldots, x_n : U_n)(y) & := & \Gamma(y) \quad (\forall i, y \neq x_i) \end{array}$$

$(\Gamma, x_1 : U_1, \ldots, x_n : U_n)$ is indeed a context and $((), x_1 : U_1, \ldots, x_n : U_n)$ is written $(x_1 : U_1, \ldots, x_n : U_n)$.

▶ **Definition 16** (Input contexts).
We write $\text{input}(\Gamma)$ if and only if for all x, we have $\text{input}(\Gamma(x))$.

▶ **Lemma 17** (Properties of contexts).
1. *$\approx$ for contexts is an equivalence relation.*
2. *$\subseteq$ for contexts is a partial pre-order and $\approx$ is its associated equivalence relation: $\Gamma \subseteq \Delta$ and $\Delta \subseteq \Gamma$ if and only if $\Gamma \approx \Delta$.*
3. *Projections: $\Gamma \cap \Delta \subseteq \Gamma$ and $\Gamma \cap \Delta \subseteq \Delta$.*
4. *Alternative definition: $\Gamma \subseteq \Delta$ if and only if there exists a context Γ' such that $\Gamma \approx \Delta \cap \Gamma'$.*
5. *Commutativity of $\cap$: $\Gamma \cap \Delta \approx \Delta \cap \Gamma$.*
6. *Associativity of $\cap$: $(\Gamma_1 \cap \Gamma_2) \cap \Gamma_3 \approx \Gamma_1 \cap (\Gamma_2 \cap \Gamma_3)$.*
7. *Stability of $\cap$: If R is either $\approx$ or $\subseteq$, $\Gamma\ R\ \Gamma'$ and $\Delta\ R\ \Delta'$, then $\Gamma \cap \Delta\ R\ \Gamma' \cap \Delta'$.*
8. *Greatest context: $\Gamma \subseteq ()$.*
9. *$(\Gamma, x : U) \subseteq \Gamma$.*
10. *If $\Gamma \subseteq \Delta$ and input(Γ), then input(Δ).*
11. *If input(Γ) and input(Δ), then input$(\Gamma \cap \Delta)$*

Proof. Straightforward. ◀

2.3 Typing system

We now have all the elements to present the typing system:

▶ **Definition 18** (Typing system).
Assume Γ is a context, M is a term, n is an integer, and U is a U-type. The judgement $\Gamma \vdash^n M : U$ is inductively defined by the rules given in Figure 4.
We write $\Gamma \vdash M : U$ if there exists n such that $\Gamma \vdash^n M : U$.

Some remarks:

- In $\Gamma \vdash^n M : U$, n is the number of uses of the rule (App_1) and it is the trivial measure on typing trees that we use.

$$\frac{}{x : F \vdash^0 x : F}(Var) \qquad \frac{\Gamma \vdash^n M : A \quad \Delta \vdash^m M : B}{\Gamma \cap \Delta \vdash^{n+m} M : A \cap B}(\cap) \qquad \frac{}{\vdash^0 M : \omega}(\omega)$$

$$\frac{\Gamma, x : U \vdash^n M : F \quad A \subseteq U}{\Gamma \vdash^n \lambda x.M : A \to F}(Fun_1) \qquad \frac{\Gamma, x : U \vdash^n M : [v] \quad \text{input}(U)}{\Gamma \vdash^n \lambda x.M : [\lambda v]}(Fun_2)$$

$$\frac{\Gamma \vdash^n M : A \to F \quad \Delta \vdash^m N : A}{\Gamma \cap \Delta \vdash^{n+m+1} MN : F}(App_1) \qquad \frac{\Gamma \vdash^n M : [k] \quad \Delta \vdash^m N : [v]}{\Gamma \cap \Delta \vdash^{n+m} MN : [kv]}(App_2)$$

Figure 4 Typing rules

- The rule (ω) is a device to simplify some definitions, proofs and properties: it is independent from the other rules and this paper could have been written without it (only with more cases in statements and proofs). In particular, $\vdash M : \omega$ gives no information about M and, for example, Theorem 24 uses A instead of U.
- (App_2) (resp. (Fun_2)) is used to type applications (resp. abstractions) that are not meant to be used in a β-reduction.
- Condition $A \subseteq U$ in rule (Fun_1) is called *subsumption* and is used to make Subject Reduction (Theorem 23) hold when the reduction is under a λ: After a subject reduction, U can turn into a different U' without changing A.
- Another advantage in having the type ω for the presentation of the typing system: Without the notation U or V, we would have to duplicate each abstraction rules that types $\lambda x.M$ (one case where $x \in \text{fv}(M)$ and one case where $x \notin \text{fv}(M)$). That would make four rules instead of two.

To define *optimality*, we first need to formally define what the *absence of subsumptions* means:

▶ **Definition 19** (No subsumptions).
In a derivation of $\Gamma \vdash^n M : U$, we say that there are no subsumptions if and only if every occurence of rule (Fun_1)

$$\frac{\Delta, x : V \vdash^n M_1 : F \quad A \subseteq V}{\Delta \vdash^n \lambda x.M_1 : A \to F}$$

is such that:
- either $A = V$
- or both $V = \omega$ and $\text{output}(A)$.

If $\Gamma \vdash^n M : U$ with no subsumptions, then we write $\Gamma \vdash^n_{\text{ns}} M : U$ and we write $\Gamma \vdash_{\text{ns}} M : U$ if there exists n such that $\Gamma \vdash^n_{\text{ns}} M : U$.

Another way of expressing the absence of subsumptions is that (Fun_1) is only used in one of the two following ways:

$$\frac{\Gamma, x : A \vdash M : F}{\Gamma \vdash \lambda x.M : A \to F} \qquad \frac{\Gamma \vdash M : F \quad x \notin \mathsf{Dom}(\Gamma)}{\Gamma \vdash \lambda x.M : [v] \to F}$$

Typing satisfies the following basic properties:

▶ **Lemma 20** (Basic properties of typing).
1. *$\Gamma \vdash^n M : U \cap V$ (resp. $\Gamma \vdash^n_{ns} M : U \cap V$) if and only if there exist Γ_1, Γ_2, n_1 and n_2 such that $\Gamma = \Gamma_1 \cap \Gamma_2$, $n = n_1 + n_2$, $\Gamma_1 \vdash^{n_1} M : U$ (resp. $\Gamma_1 \vdash^{n_1}_{ns} M : U$) and $\Gamma_2 \vdash^{n_2} M : V$ (resp. $\Gamma_2 \vdash^{n_2}_{ns} M : V$).*
2. *If $\Gamma \vdash^n M : U$ (resp. $\Gamma \vdash^n_{ns} M : U$) and $U \approx V$, then there exists Δ such that $\Gamma \approx \Delta$ and $\Delta \vdash^n M : V$ (resp. $\Delta \vdash^n_{ns} M : V$).*
3. *If $\Gamma \vdash^n M : U$ (resp. $\Gamma \vdash^n_{ns} M : U$) and $U \subseteq V$, then there exist Δ and m such that $\Gamma \subseteq \Delta$, $m \leq n$ and $\Delta \vdash^m M : V$ (resp. $\Delta \vdash^m_{ns} M : V$).*
4. *If $\Gamma \vdash M : A$, then $\mathsf{Dom}(\Gamma) = fv(M)$.*

5. *If $\Gamma \vdash M : U$, then $\mathsf{Dom}(\Gamma) \subseteq fv(M)$.*

Proof. 1. Straightforward.
2. By induction on $U \approx V$.
3. Corollary of 1 and 2.
4. By induction on $\Gamma \vdash M : A$.
5. Corollary of 4.

◀

We now have all the elements to define *optimality*: the property that a typing tree should satisfy if it should be viewed as the derivation of a big-step semantics or if we want to read from it the length of the longest β-reduction sequence (Theorems 35 and 33).

▶ **Definition 21** (Optimal typing tree).
Assume Γ is a context, n is an integer, M is a term and F is a F-type.
We write $\Gamma \vdash^n_{\text{opt}} M : F$ if and only if:
- We have $\Gamma \vdash^n_{\text{ns}} M : F$.
- We have input(Γ) and output(F).

We write $\Gamma \vdash_{\text{opt}} M : F$ if and only if there exists n such that $\Gamma \vdash^n_{\text{opt}} M : F$.

Example:

$$\dfrac{\dfrac{\dfrac{}{x : [\lambda\nabla] \vdash^0 x : [\lambda\nabla]}(Var)}{\vdash^0 \lambda x.x : [\lambda\nabla] \to [\lambda\nabla]}(Fun_1) \qquad \dfrac{\dfrac{}{y : [\nabla] \vdash^0 y : [\nabla]}(Var)}{\vdash^0 \lambda y.y : [\lambda\nabla]}(Fun_2)}{\vdash^1_{\text{opt}} (\lambda x.x)(\lambda y.y) : [\lambda\nabla]}(App_1)$$

3 Characterisation of the typing system

In Section 3.1, we prove Subject Reduction and Soundness (typable implies strongly normalising). In Section 3.2, we prove Subject Expansion and Completeness (strongly normalising implies typable).

3.1 Soundness

As usual for the proof of Subject Reduction, we first prove a substitution lemma:

▶ **Lemma 22** (Substitution lemma).
If $\Gamma, x : U \vdash^n M : A$ and $\Delta \vdash^m N : U$, then there exists Γ' such that $\Gamma' \approx \Gamma \cap \Delta$ and $\Gamma' \vdash^{n+m} M\{x := \ N\} : A$.

Proof. By induction on $\Gamma, x : U \vdash M : A$. The measure of the final typing tree is $n + m$ because, by the fact that the intersection types are non-idempotent, this proof does not do any duplications. See Appendix A. ◀

▶ **Theorem 23** (Subject Reduction).
If $\Gamma \vdash^n M : A$ and $M \longrightarrow_\beta M'$, then there exist Γ' and n' such that $\Gamma \subseteq \Gamma'$, $n > n'$ and $\Gamma' \vdash^{n'} M' : A$.

Proof. First by induction on $M \longrightarrow_\beta M'$, then by induction on A. See Appendix A. The rules (Fun_2) and (App_2) do not create any problem because we have to use the rules (Fun_1) and (App_1) to type a β-redex. ◀

In Theorem 23, we have $n > n'$ because, by the fact that types are non-idempotent, we do not do any duplications in the proof of Subject Reduction. Therefore, by Subject Reduction, for each β-reduction, the measure of the typing tree strictly decreases and then, we have Soundness as a corollary.

▶ **Theorem 24** (Soundness).
If $\Gamma \vdash^n M : A$, then $M \in \boldsymbol{SN}_{\leq n}$.

Proof. Corollary of Theorem 23: We prove by induction on n that if $\Gamma \vdash^n M : A$ then $M \in \mathbf{SN}_{\leq n}$.

Let M' be a term such that $M \longrightarrow_\beta M'$. By Theorem 23, there exist Γ' and n' such that $n' < n$ and $\Gamma' \vdash^{n'} M' : A$. By induction hypothesis, $M' \in \mathbf{SN}_{\leq n'}$. Hence, $M' \in \mathbf{SN}_{\leq n-1}$ because $n' \leq n-1$.

Therefore, $M \in \mathbf{SN}_{\leq n}$. ◀

Theorem 24 gives us a bound on the length of the longest β-reduction sequence. For a more precise result, see Theorem 33.

3.2 Completeness

We prove Subject Expansion for $\leadsto$. We could prove Subject Expansion for a larger subset of $\longrightarrow_\beta$. However, the main purpose of this paper is to emphasize the new complexity result and Theorem 35. Therefore, we prove Subject Expansion for $\leadsto$, which is enough to prove Completeness (Theorem 29).

As usual for the proof of Subject Expansion, we first prove an anti-substitution lemma:

▶ **Lemma 25** (Anti-substitution lemma).
If $\Gamma \vdash_{ns} M\{x := N\} : A$, then there exist Γ', Δ and U such that:

- $\Gamma \approx \Gamma' \cap \Delta$.
- $\Gamma', x : U \vdash_{ns} M : A$ *and* $\Delta \vdash_{ns} N : U$.
- *For all $y \notin$ Dom(Δ), $\Gamma(y) = \Gamma'(y)$.*

Proof. First by induction on M, then by induction on A. See Appendix A. The last property is necessary to preserve the absence of subsumptions in the induction. ◀

Notice that, in Lemma 25, if $x \notin \mathrm{fv}(M)$, then $U = \omega$ and we do not have any typing information on N.

In $\leadsto$, if a term is erased, then it is a normal form. Therefore, to prove Subject Expansion, we need to be able to type normal terms. This is also used in Theorem 29. To prove this, we need to know what is the type of an accumulator (when the context is an input).

▶ **Lemma 26** (Typing accumulators).
If $\Gamma \vdash M : F$, input(Γ) and acc(M), then F is of the form $[k]$.

Proof. By induction on acc(M). See Appendix A. ◀

▶ **Lemma 27** (Typing normal forms).
If M cannot be reduced by $\longrightarrow_\beta$, then there exist Γ and v such that $\Gamma \vdash_{opt} M : [v]$.

Proof. By induction on M. In particular, we use Lemma 5 and Lemma 26. See Appendix A. ◀

▶ **Theorem 28** (Subject Expansion).
If $\Gamma' \vdash_{opt} M' : F$ and $M \leadsto M'$, then there exists Γ such that $\Gamma \subseteq \Gamma'$ and $\Gamma \vdash_{opt} M : F$.

Proof. $\leadsto$ is mutually defined with $\Rrightarrow$. We therefore prove, by simultaneous induction on $M \leadsto M'$ and $M \Rrightarrow M'$, a stronger statement that forms an appropriate induction hypothesis:

If $\Gamma' \vdash_{\mathrm{ns}} M : F$ and input(Γ'), and if we are in one of the following cases:

- We have $M \Rrightarrow M'$
- We have $M \leadsto M'$ and output(F).

Then, there exists Γ such that $\Gamma \subseteq \Gamma'$, input(Γ), and $\Gamma \vdash_{\mathrm{ns}} M : F$.

See the details in Appendix A.

◀

Finally, we can prove Completeness:

▶ **Theorem 29** (Completeness).
If $M \in \boldsymbol{SN}$, then there exist Γ and F such that $\Gamma \vdash_{opt} M : F$.

Proof. By induction on the size of the longest β-reduction sequence:

- If M cannot be reduced by $\longrightarrow_\beta$: By Lemma 27, there exist Γ and F such that $\Gamma \vdash_{\text{opt}} M : F$.
- If M can be reduced by $\longrightarrow_\beta$: By Lemma 6, there exists M' such that $M \rightsquigarrow M'$. By induction hypothesis, there exist Γ' and F such that $\Gamma' \vdash_{\text{opt}} M' : F$. By Theorem 28, there exists Γ such that $\Gamma \vdash_{\text{opt}} M : F$.

◀

▶ Remark. The proof of Theorem 29 gives us an algorithm that builds an optimal typing tree from a strongly normalising term. Hence, if we consider an optimal typing tree as the derivation in a semantics, then this semantics is indeed an operational semantics, and β-reduction is only a tool to build the derivation.

This algorithm is based on the worst reduction (see Section 4.1), i.e. not very efficient. But even if there may be more efficient ways of building an typing tree for M, they have to construct, at the end of the day, something whose size is at least the length of the longest β-reduction sequence. We are therefore only interested in this as a purely theoretical construction.

Notice that we could have proved completeness (a strongly normalising term is typable) without the notion of optimality. But proving it with optimality gives more value to Theorems 33 and 35: Indeed, if a term is typable, then it is strongly normalising, so it is typable with an optimal typing tree, and therefore we can apply Theorems 33 and 35 to it.

4 Refined soundness

The purpose of this section is to prove results when we have an optimal typing tree. In Section 4.1, we prove a refined Subject Reduction property and the Complexity Result. In Section 4.2, we prove that the type of a term in an optimal typing gives the structure of its normal form, which allows us to see typing as a derivation of semantics.

4.1 Complexity

Proving the complexity result is shorter than that of [3], as we benefit from the semantical elements that we have here added to typing trees.

To prove Theorem 32 (a refined version of Theorem 23), we need a refined Substitution Lemma to preserve the absence of subsumptions:

▶ **Lemma 30** (Refined Substitution Lemma).
If $\Gamma, x : U \vdash^n_{ns} M : A$ and $\Delta \vdash^m_{ns} N : U$, then there exists Γ' such that:

- $\Gamma' \approx \Gamma \cap \Delta$.
- $\Gamma' \vdash^{n+m}_{ns} M\{x := \ N\} : A$.
- *For all $y \notin \mathit{Dom}(\Delta)$, $\Gamma(y) = \Gamma'(y)$.*

Proof. By induction on $\Gamma, x : U \vdash_{\text{ns}} M : A$. We adapt the proofs of Lemmas 22 and 25. See Appendix A. ◀

The last property of Lemma 30 has the same purpose as in Lemma 25.

One of the main advantage of this typing system compared to the one in [3] is that the measure of an optimal typing tree for a normal term is equal to zero:

▶ **Lemma 31** (Measure of normal forms).
If $\Gamma \vdash^n M : F$, M cannot be reduced by $\longrightarrow_\beta$, input(Γ) and output(F), then $n = 0$.

Proof. By induction on M with the use of Lemmas 5 and 26. Only rule (App_2) (resp. (Fun_2)) can be used to type an application (resp. an abstraction). Hence, the measure of the typing tree is indeed 0. See the details in Appendix A. ◀

In Theorem 23, the measure can decrease by more than one. However, under the assumption of optimality and using Lemma 31, we can prove a refined version of Subject Reduction:

▶ **Theorem 32** (Refined Subject Reduction).
If $\Gamma \vdash^n_{opt} M : F$ and $M \rightsquigarrow M'$, then there exists Γ' such that $\Gamma \subseteq \Gamma'$ and $\Gamma' \vdash^{n-1}_{opt} M' : F$.

Proof. As in Theorem 28, we have a stronger result with $\Rrightarrow$. We prove, by simultaneous induction on $M \rightsquigarrow M'$ and $M \Rrightarrow M'$, the following statement that forms an appropriate induction hypothesis:

If $\Gamma \vdash^n_{\mathrm{ns}} M : F$, input$(\Gamma)$, and if we are in one the following cases:

- We have $M \rightsquigarrow M'$ and output(F).
- We have $M \Rrightarrow M'$.

Then, there exists Γ' such that $\Gamma \subseteq \Gamma'$, $\Gamma' \vdash^{n-1}_{\mathrm{ns}} M' : F$ and then, by Lemma 17.10, we have input(Γ').

We adapt the proof of Theorem 23.

In particular, if $\rightsquigarrow$ erases a term, then this term is a normal term and, by Lemma 31, the measure of its typing tree is equal to zero. Also, by the fact that there are no subsumptions, we never have to discard a part of a typing tree. Moreover, by the fact that in the induction, the type of a term is not an intersection, the induction does not have to deal with the intersection rule. All of these reasons make the measure strictly decrease by one and only one. See Appendix A for details. ◀

Therefore, we can have a refined version of Theorem 24:

▶ **Theorem 33** (Complexity Result).
If $\Gamma \vdash^n_{opt} M : F$, then $M \in \boldsymbol{SN}_{=n}$.

Proof. By induction on n and by Theorem 32, and Lemmas 6 and 31, there exists a β-reduction sequence from M of length n. By Theorem 24, $M \in \mathbf{SN}_{\leq n}$. Therefore, $M \in \mathbf{SN}_{=n}$. ◀

Theorems 23 and 24 are still useful because they require a weaker hypothesis than Theorems 32 and 33. In fact, Theorems 24 and 32 are both used to prove Theorem 33.[2]

4.2 Viewing optimal typing as a big-step semantics

Section 4.1 gives us the length of the longest β-reduction sequence from an optimal typing tree. Similarly, optimality (using the refined Subject Reduction property) allows us to derive our final result: viewing the typing tree as the derivation of a big-step semantics.

We first need a result about normal forms:

▶ **Lemma 34** (Structure of a typed normal term).
If $\Gamma \vdash_{opt} M : [v]$ and M cannot be reduced by $\longrightarrow_\beta$, then struct$(M) = v$.

Proof. By induction on M: By Lemma 5, we have acc(M) or M is of the form $\lambda x.M_1$. By Definition 21, we have $\Gamma \vdash_{\mathrm{ns}} M : [v]$ and input(Γ).

- If M is of the form $\lambda x.M_1$: Then, because $[v]$ is not an arrow $A \to F$, there exist U and v_1 such that input(U), $v = \lambda v_1$ and $\Gamma, x : U \vdash_{\mathrm{ns}} M_1 : [v_1]$. Hence, input$(\Gamma, x : U)$. Therefore, $\Gamma, x : U \vdash_{\mathrm{opt}} M_1 : [v_1]$. By induction hypothesis, struct$(M_1) = v_1$. Therefore, struct$(M) = $ struct$(\lambda x.M_1) = \lambda$struct$(M_1) = \lambda v_1 = v$.

[2] We can notice that, although the results of this section are a refined version of what we prove in Section 3.1, the structure of the lemmas and proofs is closer to the ones in Section 3.2.

- If M is of the form x: Then, we have $\Gamma = (x : [v])$. Hence, we have $\mathrm{input}(x : [v])$ and $\mathrm{input}([v])$. Therefore, $v = \nabla$ and we have $\mathrm{struct}(x) = \nabla = v$.
- If M is of the form $M_1 M_2$ with $\mathrm{acc}(M_1)$: Then, we are in one of the two following cases:
  - There exist Γ_1, Γ_2 and A such that $\Gamma = \Gamma_1 \cap \Gamma_2$, $\Gamma_1 \vdash_{\mathrm{ns}} M_1 : A \to [v]$ and $\Gamma_2 \vdash_{\mathrm{ns}} M_2 : A$. By Lemma 17.10, we have $\mathrm{input}(\Gamma_1)$. By Lemma 26, $A \to [v]$ is of the form $[k]$. Contradiction.
  - There exist Γ_1, Γ_2, k and v_1 such that $\Gamma = \Gamma_1 \cap \Gamma_2$, $v = kv_1$, $\Gamma_1 \vdash_{\mathrm{ns}} M_1 : [k]$ and $\Gamma_2 \vdash_{\mathrm{ns}} M_2 : [v_1]$. By Lemma 17.10, we have $\mathrm{input}(\Gamma_1)$ and $\mathrm{input}(\Gamma_2)$. Therefore, $\Gamma_1 \vdash_{\mathrm{opt}} M_1 : [k]$ and $\Gamma_2 \vdash_{\mathrm{opt}} M_2 : [v_1]$. By induction hypothesis, $\mathrm{struct}(M_1) = k$ and $\mathrm{struct}(M_2) = v_1$. Therefore, $\mathrm{struct}(M_1 M_2) = \mathrm{struct}(M_1)\mathrm{struct}(M_2) = kv_1 = v$.

◀

▶ **Theorem 35** (Structure of the normal form of a typed term).
If $\Gamma \vdash_{opt} M : [v]$ and M' is the normal form of M, then struct$(M') = v$.

Proof. By Theorem 24, M is strongly normalising. We prove the result by induction on the longest β-reduction sequence.

- If M cannot be reduced by $\longrightarrow_\beta$, then $M = M'$ and, by Lemma 34, we can conclude.
- If M can be reduced by $\longrightarrow_\beta$, then, by Lemma 6, there exists M'' such that $M \rightsquigarrow M''$. By Theorem 32, there exists Γ'' such that $\Gamma \subseteq \Gamma''$ and $\Gamma'' \vdash_{\mathrm{opt}} M'' : [v]$. M' is also the normal form of M''. By induction hypothesis, $\mathrm{struct}(M') = v$.

◀

Theorem 35 gives us the structure of the normal form of a term. A more precise result is discussed in Section 5.2.

We can also notice that non-idempotency and the absence of subsumptions is not used to prove Theorem 35.

▶ Remark. Assume M is a closed strongly normalising term such that its normal form is a Church integer. Then, there exists a type A such that, for all closed strongly normalising terms N whose normal form is a Church integer, $\vdash_{\mathrm{opt}} N : A$ if and only if M and N are β-equivalent.[3]

We can notice that our results are concerning β-reduction/expansion/equivalence. It does not give any results on η-reduction/expansion/equivalence. Indeed, our typing system is not "η-friendly": we do not have subject reduction nor expansion for η.

5 Alternative systems

In Section 5.1, we give a simpler variant that provides the Complexity Result (still much simpler than in [3]) without giving information about the normal form. In Section 5.2, we study the issue of how to obtain a more precise information about the normal form.

5.1 Variant with no information about the normal form

This paper improves the Complexity Result of [3] and brings a new result about the normal form of a term. But if the point is only improving [3], we do not need to go as far; consider the grammar of F-types is defined as follows:

$$F, G ::= \ \nabla \ | \ \delta \ | \ A \to F$$

All the types $[kv]$ in this paper are simply collapsed into ∇, while all the types $[\lambda v]$ are collapsed into δ.

[3] This is similar to a result by Salvati [14] who provides a type system $\vdash_{\mathsf{Sal}}$ satisfiying: given a simply-typed λ-term M, there exist a context Γ and a type A such that

$$\Gamma \vdash_{\mathsf{Sal}} N : A \qquad \text{if and only if} \qquad M =_{\beta\eta} N$$

We therefore defined $\mathrm{input}(U)$ if and only if U is an intersection of ∇ and $\mathrm{output}(U)$ if and only if $U = \nabla$ or $U = \delta$.

Moreover, the rules (Fun_2) and (App_2) are defined as follows:

$$\frac{\Gamma, x : U \vdash^n M : F \quad \mathrm{input}(U) \quad \mathrm{output}(F)}{\Gamma \vdash^n \lambda x.M : \delta} \qquad \frac{\Gamma_1 \vdash^{n_1} M_1 : \nabla \quad \Gamma_2 \vdash^{n_2} M_2 : F \quad \mathrm{output}(F)}{\Gamma_1 \cap \Gamma_2 \vdash^{n_1+n_2} M_1M_2 : \nabla}$$

The proofs of Soundness, Completeness and the Complexity Result are similar.

5.2 Obtaining the exact normal form

It would be interesting to enrich the typing system of this paper to improve Theorem 35, so that the type of a term gives the exact normal form instead of just its structure. To highlight the separation between terms and types, even when these denote terms in normal form, we use a different syntax in the enriched grammar for v and k:

$$\begin{array}{lcl} v & ::= & \lambda\mathfrak{x}.M \mid k \\ k & ::= & \mathfrak{x} \mid kv \end{array}$$

where $\mathfrak{x}, \mathfrak{y}, \mathfrak{z}$ are *labels*, which differ from λ-term variables in that we choose them to not be α-convertible in v and k (see below why). A naive way of modifying (Fun_2) is:

$$\frac{\Gamma, x : U \vdash^n M : [v] \quad \mathrm{input}_{\mathfrak{y}}(U)}{\Gamma \vdash^n \lambda x.M : [\lambda\mathfrak{y}.v]}$$

where $\mathrm{input}_{\mathfrak{x}}(U)$ means that U is an intersection of $[\mathfrak{x}]$. Now in order to give an optimal typing tree to $(\lambda x.\lambda y.xy)(\lambda z.z)$, indicating its normal form with the type $[\lambda\mathfrak{z}.\mathfrak{z}]$, we would like this instance of (Fun_2):

$$\frac{x : [\mathfrak{z}] \to [\mathfrak{z}], y : [\mathfrak{z}] \vdash xy : [\mathfrak{z}]}{x : [\mathfrak{z}] \to [\mathfrak{z}] \vdash \lambda y.xy : [\lambda\mathfrak{z}.\mathfrak{z}]}$$

which shows the need to not consider $\lambda\mathfrak{z}.\mathfrak{z}$ to be α-convertible (to $\lambda\mathfrak{y}.\mathfrak{y}$, for instance). Yet, avoiding name clashes and degeneration of types prompts for a side-condition for (Fun_2) of the form $\mathfrak{y} \notin \mathrm{fv}(\Gamma)$ (freshness of $\mathfrak{y}$), which unfortunately forbids the above example (so we would not have Completeness, for lack of Subject Expansion).

To solve this problem we need a subtler way of defining the freshness of $\mathfrak{y}$ in (Fun_2). We add a condition in the definition of an optimal typing tree: In the typing tree, a label $\mathfrak{y}$ must be used at most once in an occurrence of (Fun_2). This is a notion of *global freshness*, that is somewhat similar to Barendregt's convention for labels, and related to the idea that principal types can specify normal forms [13]. This condition can blend in the lemmas and theorems of the paper:

- Completeness: We have enough fresh labels to type a normal term when needed.
- Soundness: Subject Reduction does not do any duplications (because of non-idempotency), and therefore, global freshness is preserved by (refined or not) Subject Reduction.

More formally we need to collect the labels $\mathfrak{y}$ used in (Fun_2). Therefore, we have typing judgements of the form $\Gamma \vdash^n_{\mathfrak{A}} MU$ with $\mathfrak{A}$ a finite multi-set of labels. This makes the lemmas and proofs harder to read and this is the reason we have chosen to keep the version with structures as the main theory.

Alternatively, we could work in a calculus that does not have λ-abstractions. For example, the calculus with only combinators (S, K, etc ...) and applications would be a good candidate.

6 Conclusion

We have presented a typing system of non-idempotent intersection types that characterises strongly normalising λ-terms and in which an (optimal) typing derivation provides

1. the exact length of the longest β-reduction sequence
2. the structure of the normal form

The typing system is an enhanced version of that introduced in [3] and [4], which improves and simplifies the result similar to 1. that was proved in [3]. Indeed, we used fewer definitions and lemmas, and the measure of an optimal typing tree is exactly the length of the longest β-reduction sequence from the typed term.

Perhaps more importantly, introducing information about normal forms, within types, has turned the typing system into a system deriving a big-step semantics for strongly normalising terms (provided we accept the structure of normal forms as an acceptable semantics). This style of derivations (based on typing) is unusual in that it never computes a substitution, and it relates to worst-case reduction strategies (generating longest β-reduction sequences).

We could adapt this improvement to a calculus with constructors and a fixpoint operator in order to obtain these results in a more concrete and usable calculus.

References

1 Fabio Alessi, Franco Barbanera, and Mariangiola Dezani-Ciancaglini. Intersection types and lambda models. *Theoret. Comput. Sci.*, 355(2):108–126, 2006.

2 Henk Barendregt, Mario Coppo, and Mariangiola Dezani-Ciancaglini. A filter lambda model and the completeness of type assignment. *J. of Symbolic Logic*, 48(4):931–940, 1983.

3 Alexis Bernadet and Stéphane Lengrand. Complexity of strongly normalising λ-terms via non-idempotent intersection types. In Martin Hofmann, editor, *Proc. of the 14th Int. Conf. on Foundations of Software Science and Computation Structures (FOSSACS'11)*, volume 6604 of *LNCS*. Springer-Verlag, March 2011.

4 Alexis Bernadet and Stéphane Lengrand. Filter models: non-idempotent intersection types, orthogonality and polymorphism. In Marc Bezem, editor, *Proc. of the 20th Annual Conf. of the European Association for Computer Science Logic (CSL'11)*, LIPIcs. Schloss Dagstuhl LCI, September 2011.

5 M. Coppo and M. Dezani-Ciancaglini. A new type assignment for lambda-terms. *Archiv für mathematische Logik und Grundlagenforschung*, 19:139–156, 1978.

6 Daniel de Carvalho. Intersection types for light affine lambda calculus. *ENTCS*, 136:133–152, 2005.

7 Daniel de Carvalho. Execution time of lambda-terms via denotational semantics and intersection types. *CoRR*, abs/0905.4251, 2009.

8 Mariangiola Dezani-Ciancaglini, Silvia Ghilezan, and Silvia Likavec. Behavioural Inverse Limit Models. *Theoret. Comput. Sci.*, 316(1–3):49–74, 2004.

9 Mariangiola Dezani-Ciancaglini, Furio Honsell, and Yoko Motohama. Compositional characterisations of *lambda*-terms using intersection types. *Theoret. Comput. Sci.*, 340(3):459–495, 2005.

10 J. Gallier. Typing untyped lambda terms, or reducibility strikes again. *Ann. Pure Appl. Logic*, 91:231–270, 1998.

11 Jean-Yves Girard. Linear logic. *Theoret. Comput. Sci.*, 50(1):1–101, 1987.

12 D. Leivant. Typing and computational properties of lambda expressions. *Theoret. Comput. Sci.*, 44(1):51–68, 1986.

13 Peter Møller Neergaard and Harry G. Mairson. Types, potency, and idempotency: why nonlinearity and amnesia make a type system work. In Chris Okasaki and Kathleen Fisher, editors, *Proc. of the 9th ACM Intern. Conf. on Functional Programming*, pages 138–149. ACM Press, September 2004.

14 Sylvain Salvati. On the Membership Problem for Non-Linear Abstract Categorial Grammars. *J. of Logic, Language, and Information*, 19(2):163–183, April 2010.

15 Dana S. Scott. Domains for denotational semantics. In Mogens Nielsen and Erik Meineche Schmidt, editors, *Proc. of the 9th Intern. Col. on Automata, Languages and Programming (ICALP)*, volume 140 of *LNCS*, pages 577–613. Springer-Verlag, July 1982.

16 S. van Bakel. Intersection Type Assignment Systems. *Theoret. Comput. Sci.*, 151(2):385–435, 1995.

A Full proofs

▶ **Lemma 5** (Shape of a normal form). *If M cannot be reduced by $\longrightarrow_\beta$, then*

- *either we have acc(M),*
- *or M is of the form $\lambda x.M_1$.*

Proof. By induction on M.

- If M is of the form $\lambda x.M_1$, then we can conclude.
- If M is a variable x, then we have $\mathrm{acc}(M)$.
- If M is of the form M_1M_2: Then, M_1 cannot be reduced by $\longrightarrow_\beta$. By induction hypothesis on M_1, either we have $\mathrm{acc}(M_1)$ or M_1 is of the form $\lambda x.M_3$.
  - If $\mathrm{acc}(M_1)$, then $\mathrm{acc}(M_1M_2)$.
  - If M_1 is of the form $\lambda x.M_3$, then $M \longrightarrow_\beta M_3\{x := M_2\}$. Contradiction.

◀

▶ **Lemma 6** (Applicability of $\rightsquigarrow$). *If M can be reduced by $\longrightarrow_\beta$, then M can be reduced by $\rightsquigarrow$.*

Proof. We prove by induction on M that if M can be reduced by $\longrightarrow_\beta$ then:

- If M is of the form $\lambda x.M_1$, then there exists M' such that $M \rightsquigarrow M'$.
- If not, then there exists M' such $M \Rightarrow M'$.

Therefore, in both cases there exists M' such that $M \rightsquigarrow M'$.

- If M is a variable, then M cannot be reduced by $\longrightarrow_\beta$. Contradiction.
- If M is of the form $\lambda x.M_1$: Then, M_1 can be reduced by $\longrightarrow_\beta$. By induction hypothesis, there exists M_1' such that $M_1 \rightsquigarrow M_1'$. Therefore, $M \rightsquigarrow \lambda x.M_1'$.
- If M is of the form M_1M_2: Therefore, we are in one of the following cases:
  - M_1 is of the form $\lambda x.M_3$ and $x \in \mathrm{fv}(M_3)$: Therefore, $M \Rightarrow M_3\{x := M_2\}$.
  - M_1 is of the form $\lambda x.M_3$, $x \notin \mathrm{fv}(M_3)$ and M_2 can be reduced by $\longrightarrow_\beta$: By induction hypothesis, there exist M_2' such that $M_2 \rightsquigarrow M_2'$. Therefore $M \Rightarrow (\lambda x.M_3)M_2'$.
  - M_1 is of the form $\lambda x.M_3$, $x \notin \mathrm{fv}(M_3)$ and M_2 cannot be reduced by $\longrightarrow_\beta$: Then, $M \Rightarrow M_3$.
  - M_1 is not of the form $\lambda x.M_3$ and M_1 can be reduced by $\longrightarrow_\beta$: By induction hypothesis, there exist M_1' such that $M_1 \Rightarrow M_1'$. Therefore, $M \Rightarrow M_1'M_2$.
  - M_1 is not of the form $\lambda x.M_3$ and M_1 cannot be reduced by $\longrightarrow_\beta$: By the fact that M_1 is not of the form $\lambda x.M_3$, M_2 can be reduced by $\longrightarrow_\beta$. By induction hypothesis, there exist M_2' such that $M_2 \rightsquigarrow M_2'$. By Lemma 5, we have $\mathrm{acc}(M_1)$. Therefore, $M \Rightarrow M_1M_2'$.

◀

▶ **Lemma 8** (Structure of a normal term). *M cannot be reduced by $\longrightarrow_\beta$ if and only if struct(M) is well-defined. Moreover, if we have acc(M), then struct(M) is of the form k.*

Proof. By induction on M.

- If M is of the form x: x cannot be reduced by $\longrightarrow_\beta$, $\mathrm{struct}(x)$ is well-defined and $\mathrm{struct}(x) = \nabla$ which is of the form k.
- If M is of the form $\lambda x.M_1$:
  - If $\lambda x.M_1$ cannot be reduced by $\longrightarrow_\beta$: Then, M_1 cannot be reduced by $\longrightarrow_\beta$. By induction hypothesis, $\mathrm{struct}(M_1)$ is well-defined. Therefore, $\mathrm{struct}(\lambda x.M_1)$ is well-defined and we do not have $\mathrm{acc}(\lambda x.M_1)$.
  - If $\mathrm{struct}(\lambda x.M_1)$ is well-defined: Then, $\mathrm{struct}(M_1)$ is well-defined. By induction hypothesis, M_1 cannot be reduced by $\longrightarrow_\beta$. Therefore, $\lambda x.M_1$ cannot be reduced by $\longrightarrow_\beta$.

- If M is of the form M_1M_2:
  - If M_1M_2 cannot be reduced by $\longrightarrow_\beta$: Then, M_1 and M_2 cannot be reduced by $\longrightarrow_\beta$. By induction hypothesis, $\mathrm{struct}(M_1)$ and $\mathrm{struct}(M_2)$ are well-defined. By Lemma 5, we have $\mathrm{acc}(M_1)$ or M_1 is of the form $\lambda x.M_3$. If M_1 is of the form $\lambda x.M_3$, then $M_1M_2 \longrightarrow_\beta M_3\{x := M_2\}$. Contradiction. Hence, we have $\mathrm{acc}(M_1)$. By induction hypothesis, $\mathrm{struct}(M_1)$ is of the form k. Therefore, $\mathrm{struct}(M_1M_2)$ is well-defined and $\mathrm{struct}(M_1M_2) = \mathrm{struct}(M_1)\mathrm{struct}(M_2)$ which is of the form k'.
  - If $\mathrm{struct}(M_1M_2)$ is well-defined: Then, $\mathrm{struct}(M_1)$ and $\mathrm{struct}(M_2)$ are well-defined, and $\mathrm{struct}(M_1)$ is of the form k. By induction hypothesis, M_1 and M_2 cannot be reduced by $\longrightarrow_\beta$. If M_1 is of the form $\lambda x.M_3$, then $\mathrm{struct}(M_1) = \lambda\mathrm{struct}(M_3)$ which is not of the form k. Contradiction. Hence, M_1 is not of the form $\lambda x.M_3$. Therefore, M_1M_2 cannot be reduced by $\longrightarrow_\beta$.

◀

▶ **Lemma 12** (Properties of $\approx$).

1. *Neutrality of ω: $U \cap \omega = \omega \cap U = U$.*
2. *Strictness of F-types: If $U \approx F$, then $U = F$.*
3. *Strictness of ω: If $U \approx \omega$, then $U = \omega$.*
4. *$\approx$ is an equivalence relation.*
5. *Commutativity of $\cap$: $U \cap V \approx V \cap U$*
6. *Associativity of $\cap$: $U_1 \cap (U_2 \cap U_3) \approx (U_1 \cap U_2) \cap U_3$*
7. *Stability of $\cap$: If $U \approx U'$ and $V \approx V'$, then $U \cap V \approx U' \cap V'$.*
8. *If $U \cap V = \omega$, then $U = V = \omega$.*
9. *If $U \cap V \approx U$, then $V = \omega$.*
10. *If $U \approx V$ and $\mathrm{input}(U)$, then $\mathrm{input}(V)$.*
11. *$\mathrm{input}(U \cap V)$ if and only if $\mathrm{input}(U)$ and $\mathrm{input}(V)$.*

Proof. **1.** Straightforward.

2. By induction on $U \approx F$.

3. Straightforward.

4.

- Reflexivity: We prove that $U \approx U$ by induction on U.
- Symmetry: We prove by induction on $U \approx V$ that if $U \approx V$, then $V \approx U$.
- Transitivity: Straightforward.

5. Straightforward.

6. Straightforward.

7. Straightforward.

8. Straightforward.

9. For all U, we construct $\varphi(U)$ defined by induction on U as follows:

$$\begin{array}{lcl} \varphi(F) & := & 1 \\ \varphi(A \cap B) & := & \varphi(A) + \varphi(B) \\ \varphi(\omega) & := & 0 \end{array}$$

By induction on U, if $\varphi(U) = 0$, then $U = \omega$
We also have $\varphi(U \cap V) = \varphi(U) + \varphi(V)$.
By induction on $U \approx V$, if $U \approx V$, then $\varphi(U) = \varphi(V)$.
If $U \cap V \approx U$: Then, $\varphi(U \cap V) = \varphi(U)$. Therefore, $\varphi(U) + \varphi(V) = \varphi(U)$. Hence, $\varphi(V) = 0$. Therefore, $V = \omega$.

10. By induction on $U \approx V$.

11. Straightforward.

◀

▶ **Lemma 22** (Substitution lemma).
If $\Gamma, x : U \vdash^n M : A$ *and* $\Delta \vdash^m N : U$*, then there exists* Γ' *such that* $\Gamma' \approx \Gamma \cap \Delta$ *and* $\Gamma' \vdash^{n+m} M\{x := N\} : A$*.*

Proof. By induction on $\Gamma, x : U \vdash^n M : A$.

- For $\dfrac{}{x : F \vdash^0 x : F}$ with $\Gamma = ()$, $n = 0$, $M = x$, $U = F$ and $A = F$: We have $x\{x := N\} = N$. By hypothesis, $\Delta \vdash^m N : U$. Therefore, $\Delta \vdash^m M\{x := N\} : F$ with $n + m = m$ and $\Gamma \cap \Delta = () \cap \Delta = \Delta$.
- For $\dfrac{}{y : F \vdash^0 y : F}$ with $y \neq x$, $\Gamma = (y : F)$, $n = 0$, $M = y$, $U = \omega$, and $A = F$: By hypothesis, $\Delta \vdash^m N : \omega$. Hence, $\Delta = ()$ and $m = 0$. We have $y\{x := N\} = y$. Therefore, $y : F \vdash^0 M\{x := N\} : F$ with $n+m = 0$ and $\Gamma \cap \Delta = (y : F) \cap () = (y : F)$.
- For $\dfrac{\Gamma_1, x : U_1 \vdash^{n_1} M : A_1 \quad \Gamma_2, x : U_2 \vdash^{n_2} M : A_2}{\Gamma_1 \cap \Gamma_2, x : U_1 \cap U_2 \vdash^{n_1+n_2} M : A_1 \cap A_2}$ with $\Gamma = \Gamma_1 \cap \Gamma_2$, $n = n_1 + n_2$, $U = U_1 \cap U_2$ and $A = A_1 \cap A_2$: By hypothesis, $\Delta \vdash^m N : U_1 \cap U_2$. By Lemma 20.1, there exist Δ_1, Δ_2, m_1 and m_2 such that $\Delta = \Delta_1 \cap \Delta_2$, $m = m_1 + m_2$, $\Delta_1 \vdash^{m_1} N : U_1$ and $\Delta_2 \vdash^{m_2} N : U_2$. By induction hypothesis, there exist Γ'_1 and Γ'_2 such that $\Gamma'_1 \approx \Gamma_1 \cap \Delta_1$, $\Gamma'_2 \approx \Gamma_2 \cap \Delta_2$, $\Gamma'_1 \vdash^{n_1+m_1} M\{x := N\} : A_1$ and $\Gamma'_2 \vdash^{n_2+m_2} M\{x := N\} : A_2$. Therefore, $\Gamma'_1 \cap \Gamma'_2 \vdash^{n_1+m_1+n_2+m_2} M\{x := N\} : A_1 \cap A_2$ with $\Gamma'_1 \cap \Gamma'_2 \approx (\Gamma_1 \cap \Delta_1) \cap (\Gamma_2 \cap \Delta_2) \approx (\Gamma_1 \cap \Gamma_2) \cap (\Delta_1 \cap \Delta_2) = \Gamma \cap \Delta$ and $n_1 + m_1 + n_2 + m_2 = n + m$.
- For $\dfrac{\Gamma, x : U, y : V \vdash^n M_1 : F \quad B \subseteq V}{\Gamma, x : U \vdash^n \lambda y.M_1 : B \to F}$ with $M = \lambda y.M_1$, $x \neq y$, $y \notin \mathrm{fv}(N)$ and $A = B \to F$. We have $(\lambda y.M_1)\{x := N\} = \lambda y.M_1\{x := N\}$. By induction hypothesis, there exists Γ' such that $\Gamma' \approx (\Gamma, y : V) \cap \Delta$ and $\Gamma' \vdash^{n+m} M_1\{y := N\} : F$. By Lemma 20.5, $\mathsf{Dom}(\Delta) \subseteq \mathrm{fv}(N)$. Therefore, $y \notin \mathsf{Dom}(\Delta)$ and $(\Gamma, y : V) \cap \Delta = (\Gamma \cap \Delta, y : V)$. There exist a unique Γ'' and a unique V' such that $(\Gamma'', y : V') = \Gamma'$. Therefore, $\Gamma'' \approx \Gamma \cap \Delta$ and $V \approx V'$. Hence, $B \subseteq V'$. Therefore, $\Gamma'' \vdash^{n+m} \lambda y.M_1\{x := N\} : B \to F$.
- For $\dfrac{\Gamma, x : U, y : V \vdash^n M_1 : [v] \quad \mathrm{input}(V)}{\Gamma, x : U \vdash^n \lambda y.M_1 : [\lambda v]}$ with $M = \lambda y.M_1$, $y \notin \mathrm{fv}(N)$, $A = [\lambda v]$ and $y \neq x$: We have $(\lambda y.M_1)\{x := N\} = \lambda y.M_1\{x := N\}$. By induction hypothesis, there exists Γ' such that $\Gamma' \approx (\Gamma, y : V) \cap \Delta$ and $\Gamma' \vdash^{n+m} M_1\{x := N\} : [v]$. By Lemma 20.5, $\mathsf{Dom}(\Delta) \subseteq \mathrm{fv}(N)$. Therefore, $y \notin \mathsf{Dom}(\Delta)$ and $(\Gamma, y : V) \cap \Delta = (\Gamma \cap \Delta, y : V)$. There exist a unique Γ'' and a unique V' such that $(\Gamma'', y : V') = \Gamma'$. Therefore, $\Gamma'' \approx \Gamma \cap \Delta$ and $V \approx V'$. By Lemma 12.10, we have $\mathrm{input}(V')$. Therefore, $\Gamma'' \vdash^{n+m} \lambda y.M_1\{x := N\} : [\lambda v]$.
- For $\dfrac{\Gamma_1, x : U_1 \vdash^{n_1} M_1 : B \to F \quad \Gamma_2, x : U_2 \vdash^{n_2} M_2 : B}{\Gamma_1 \cap \Gamma_2, x : U_1 \cap U_2 \vdash^{n_1+n_2+1} M_1 M_2 : F}$ with $\Gamma = \Gamma_1 \cap \Gamma_2$, $n = n_1 + n_2 + 1$, $U = U_1 \cap U_2$, $M = M_1 M_2$ and $A = F$: We have $(M_1 M_2)\{x := N\} = M_1\{x := N\} M_2\{x := N\}$. By hypothesis, $\Delta \vdash^m N : U_1 \cap U_2$. By Lemma 20.1, there exist Δ_1, Δ_2, m_1 and m_2 such that $\Delta = \Delta_1 \cap \Delta_2$, $m = m_1 + m_2$, $\Delta_1 \vdash^{m_1} N : U_1$ and $\Delta_2 \vdash^{m_2} N : U_2$. By induction hypothesis, there exist Γ'_1 and Γ'_2 such that $\Gamma'_1 \approx \Gamma_1 \cap \Delta_1$, $\Gamma'_2 \approx \Gamma_2 \cap \Delta_2$, $\Gamma'_1 \vdash^{n_1+m_1} M_1\{x := N\} : B \to F$ and $\Gamma'_2 \vdash^{n_2+m_2} M_2\{x := N\} : B$. Therefore, $\Gamma'_1 \cap \Gamma'_2 \vdash^{n_1+m_1+n_2+m_2+1} M_1 M_2 : F$ with $\Gamma'_1 \cap \Gamma'_2 \approx (\Gamma_1 \cap \Delta_1) \cap (\Gamma_2 \cap \Delta_2) \approx (\Gamma_1 \cap \Gamma_2) \cap (\Delta_1 \cap \Delta_2) = \Gamma \cap \Delta$ and $n_1 + m_1 + n_2 + m_2 + 1 = n + m$.
- For $\dfrac{\Gamma_1, x : U_1 \vdash^{n_1} M_1 : [k] \quad \Gamma_2, x : U_2 \vdash^{n_2} M_2 : [v]}{\Gamma_1 \cap \Gamma_2, x : U_1 \cap U_2 \vdash^{n_1+n_2} M_1 M_2 : [kv]}$ with $\Gamma = \Gamma_1 \cap \Gamma_2$, $n = n_1 + n_2$, $U = U_1 \cap U_2$, $M = M_1 M_2$ and $A = [kv]$: We have $(M_1 M_2)\{x := N\} = M_1\{x := N\} M_2\{x := N\}$. By hypothesis, $\Delta \vdash^m N : U_1 \cap U_2$. By Lemma 20.1, there exist Δ_1, Δ_2, m_1 and m_2 such that $\Delta = \Delta_1 \cap \Delta_2$, $m = m_1 + m_2$, $\Delta_1 \vdash^{m_1} N : U_1$ and $\Delta_2 \vdash^{m_2} N : U_2$. By induction hypothesis, there exist Γ'_1 and Γ'_2 such that $\Gamma'_1 \approx \Gamma_1 \cap \Delta_1$, $\Gamma'_2 \approx \Gamma_2 \cap \Delta_2$, $\Gamma'_1 \vdash^{n_1+m_1} M_1\{x := N\} : [k]$ and $\Gamma'_2 \vdash^{n_2+m_2} M_2\{x := N\} : [v]$. Therefore, $\Gamma'_1 \cap \Gamma'_2 \vdash^{n_1+m_1+n_2+m_2} M_1 M_2 : [kv]$ with $\Gamma'_1 \cap \Gamma'_2 \approx (\Gamma_1 \cap \Delta_1) \cap (\Gamma_2 \cap \Delta_2) \approx (\Gamma_1 \cap \Gamma_2) \cap (\Delta_1 \cap \Delta_2) = \Gamma \cap \Delta$ and $n_1 + m_1 + n_2 + m_2 = n + m$.

◀

▶ **Theorem 23** (Subject Reduction).
If $\Gamma \vdash^n M : A$ and $M \longrightarrow_\beta M'$, then there exist Γ' and n' such that $\Gamma \subseteq \Gamma'$, $n > n'$ and $\Gamma' \vdash^{n'} M' : A$.

Proof. First by induction on $M \longrightarrow_\beta M'$, then by induction on A.

- If A is of the form $A_1 \cap A_2$: Then, there exist Γ_1, Γ_2, n_1 and n_2 such that $\Gamma = \Gamma_1 \cap \Gamma_2$, $n = n_1 + n_2$, $\Gamma_1 \vdash^{n_1} M : A_1$ and $\Gamma_2 \vdash^{n_2} M : A_2$. By induction hypothesis on $(M \longrightarrow_\beta M', A_1)$ and $(M \longrightarrow_\beta M', A_2)$, there exist Γ'_1, Γ'_2, n'_1 and n'_2 such that $\Gamma_1 \subseteq \Gamma'_1$, $\Gamma_2 \subseteq \Gamma'_2$, $n_1 > n'_1$, $n_2 > n'_2$, $\Gamma'_1 \vdash^{n'_1} M' : A_1$ and $\Gamma'_2 \vdash^{n'_2} M' : A_2$. Therefore, $\Gamma'_1 \cap \Gamma'_2 \vdash^{n'_1+n'_2} M' : A_1 \cap A_2$ with $\Gamma = \Gamma_1 \cap \Gamma_2 \subseteq \Gamma'_1 \cap \Gamma'_2$ and $n = n_1 + n_2 > n'_1 + n'_2$.
- For $\dfrac{}{(\lambda x.M_1)M_2 \longrightarrow_\beta M_1\{x := M_2\}}$ with $M = (\lambda x.M_1)M_2$ and A is of the form F:
  Then, we are in one of the following cases:
  - There exist Γ_1, Γ_2, v and k such that $\Gamma = \Gamma_1 \cap \Gamma_2$, $\Gamma_1 \vdash \lambda x.M_1 : [k]$, $\Gamma_2 \vdash M_2 : [v]$ and $F = [kv]$. An abstraction $\lambda x.M_1$ cannot have $[k]$ as a type (it is either an arrow $B \to G$ or of the form $[\lambda v_1]$). Contradiction.
  - There exist Γ_1, Γ_2, n_1, n_2, B such that $\Gamma = \Gamma_1 \cap \Gamma_2$, $n = n_1 + n_2 + 1$, $\Gamma_1 \vdash^{n_1} \lambda x.M_1 : B \to F$ and $\Gamma_2 \vdash^{n_2} M_2 : B$: Then, there exists U such that $B \subseteq U$ and $\Gamma_1, x : U \vdash^{n_1} M_1 : F$. By Lemma 20.3, there exist Γ'_2 and n'_2 such that $\Gamma_2 \subseteq \Gamma'_2$, $n_2 \geq n'_2$ and $\Gamma'_2 \vdash^{n'_2} M_2 : U$. By Lemma 22, there exists Γ' such that $\Gamma' \approx \Gamma_1 \cap \Gamma'_2$ and $\Gamma' \vdash^{n_1+n'_2} M_1\{x := M_2\} : F$ with $\Gamma = \Gamma_1 \cap \Gamma_2 \subseteq \Gamma_1 \cap \Gamma'_2 \approx \Gamma'$ and $n = n_1 + n_2 + 1 > n_1 + n_2 \geq n_1 + n'_2$.
- For $\dfrac{M_1 \longrightarrow_\beta M'_1}{\lambda x.M_1 \longrightarrow_\beta \lambda x.M'_1}$ with $M = \lambda x.M_1$ and A is of the form F, we are in one of the following cases:
  - There exist B, G and U such that $F = B \to G$, $B \subseteq U$ and $\Gamma, x : U \vdash^n M_1 : G$. By induction hypothesis, there exists Γ'_1 and n' such that $(\Gamma, x : U) \subseteq \Gamma'_1$, $n > n'$ and $\Gamma'_1 \vdash^{n'} M'_1 : G$. There exist a unique Γ' and a unique U' such that $\Gamma'_1 = (\Gamma', x : U')$. Therefore, $\Gamma \subseteq \Gamma'$ and $U \subseteq U'$. Hence, $B \subseteq U'$. Therefore $\Gamma' \vdash^{n'} \lambda x.M'_1 : B \to G$.
  - There exist U and v such that $\text{input}(U)$, $F = [\lambda v]$ and $\Gamma, x : U \vdash^n M_1 : [v]$. By induction hypothesis, there exist Γ'_1 and n' such that $(\Gamma, x : U) \subseteq \Gamma'_1$, $n > n'$ and $\Gamma'_1 \vdash^{n'} M'_1 : [v]$. There exist a unique Γ' and a unique U' such that $\Gamma'_1 = (\Gamma', x : U')$. Therefore, $\Gamma \subseteq \Gamma'$ and $U \subseteq U'$. Hence, by Lemma 14.5, we have $\text{input}(U')$. Therefore, $\Gamma' \vdash^{n'} \lambda x.M'_1 : [\lambda v]$.
- For $\dfrac{M_1 \longrightarrow_\beta M'_1}{M_1 M_2 \longrightarrow_\beta M'_1 M_2}$ with $M = M_1 M_2$ and A is of the form F, we are in one of the following cases:
  - There exist Γ_1, Γ_2, n_1, n_2 and B such that $\Gamma = \Gamma_1 \cap \Gamma_2$, $n = n_1 + n_2 + 1$, $\Gamma_1 \vdash^{n_1} M_1 : B \to F$ and $\Gamma_2 \vdash^{n_2} M_2 : B$. By induction hypothesis, there exist Γ'_1 and n'_1 such that $\Gamma_1 \subseteq \Gamma'_1$, $n_1 > n'_1$ and $\Gamma'_1 \vdash^{n'_1} M'_1 : B \to F$. Therefore, $\Gamma'_1 \cap \Gamma_2 \vdash^{n'_1+n_2+1} M'_1 M_2 : F$ with $\Gamma = \Gamma_1 \cap \Gamma_2 \subseteq \Gamma'_1 \cap \Gamma_2$ and $n = n_1 + n_2 + 1 > n'_1 + n_2 + 1$.
  - There exist Γ_1, Γ_2, n_1, n_2, k and v such that $\Gamma = \Gamma_1 \cap \Gamma_2$, $n = n_1 + n_2$, $F = [kv]$, $\Gamma_1 \vdash^{n_1} M_1 : [k]$ and $\Gamma_2 \vdash^{n_2} M_2 : [v]$. By induction hypothesis, there exist Γ'_1 and n'_1 such that $\Gamma_1 \subseteq \Gamma'_1$, $n_1 > n'_1$ and $\Gamma'_1 \vdash^{n'_1} M'_1 : [k]$. Therefore $\Gamma'_1 \cap \Gamma_2 \vdash^{n'_1+n_2} M'_1 M_2 : [kv]$ with $\Gamma = \Gamma_1 \cap \Gamma_2 \subseteq \Gamma'_1 \cap \Gamma_2$ and $n = n_1 + n_2 > n'_1 + n_2$.
- For $\dfrac{M_2 \longrightarrow_\beta M'_2}{M_1 M_2 \longrightarrow_\beta M_1 M'_2}$ with $M = M_1 M_2$ and A is of the form F, we are in one of the following cases:
  - There exist Γ_1, Γ_2, n_1, n_2 and B such that $\Gamma = \Gamma_1 \cap \Gamma_2$, $n = n_1 + n_2 + 1$, $\Gamma_1 \vdash^{n_1} M_1 : B \to F$ and $\Gamma_2 \vdash^{n_2} M_2 : B$. By induction hypothesis, there exist Γ'_2 and

n'_2 such that $\Gamma_2 \subseteq \Gamma'_2$, $n_2 > n'_2$ and $\Gamma'_2 \vdash^{n'_2} M'_2 : B$. Therefore, $\Gamma_1 \cap \Gamma'_2 \vdash^{n_1+n'_2+1} M_1M'_2 : F$ with $\Gamma = \Gamma_1 \cap \Gamma_2 \subseteq \Gamma_1 \cap \Gamma'_2$ and $n = n_1 + n_2 + 1 > n_1 + n'_2 + 1$.

- There exist Γ_1, Γ_2, n_1, n_2, k and v such that $\Gamma = \Gamma_1 \cap \Gamma_2$, $n = n_1 + n_2$, $F = [kv]$, $\Gamma_1 \vdash^{n_1} M_1 : [k]$ and $\Gamma_2 \vdash^{n_2} M_2 : [v]$. By induction hypothesis, there exist Γ'_2 and n'_2 such that $\Gamma_2 \subseteq \Gamma'_2$, $n_2 > n'_2$ and $\Gamma'_2 \vdash^{n'_2} M'_2 : [v]$. Therefore, $\Gamma_1 \cap \Gamma'_2 \vdash^{n_1+n'_2} M_1M'_2 : [kv]$ with $\Gamma = \Gamma_1 \cap \Gamma_2 \subseteq \Gamma_1 \cap \Gamma'_2$ and $n = n_1 + n_2 > n_1 + n'_2$.

◀

▶ **Lemma 25** (Anti-substitution lemma).
If $\Gamma \vdash_{ns} M\{x := N\} : A$, then there exist Γ', Δ and U such that:
- $\Gamma \approx \Gamma' \cap \Delta$.
- $\Gamma', x : U \vdash_{ns} M : A$ *and* $\Delta \vdash_{ns} N : U$.
- *For all* $y \notin \mathsf{Dom}(\Delta)$, $\Gamma(y) = \Gamma'(y)$.

Proof. First by induction on M, then by induction on A.
- If A is of the form $A_1 \cap A_2$: Then, there exist Γ_1 and Γ_2 such that $\Gamma = \Gamma_1 \cap \Gamma_2$, $\Gamma_1 \vdash_{\text{ns}} M\{x := N\} : A_1$ and $\Gamma_2 \vdash_{\text{ns}} M\{x := N\} : A_2$. By induction hypothesis on (M, A_1) and (M, A_2), there exist Γ'_1, Γ'_2, Δ_1, Δ_2, U_1 and U_2 such that:
  - $\Gamma_1 \approx \Gamma'_1 \cap \Delta_1$ and $\Gamma_2 \approx \Gamma'_2 \cap \Delta_2$.
  - $\Gamma'_1, x : U_1 \vdash_{\text{ns}} M : A_1$, $\Gamma'_2, x : U_2 \vdash_{\text{ns}} M : A_2$, $\Delta_1 \vdash_{\text{ns}} N : U_1$ and $\Delta_2 \vdash_{\text{ns}} N : U_2$.
  - For all $y \notin \mathsf{Dom}(\Delta_1)$, $\Gamma_1(y) = \Gamma'_1(y)$.
  - For all $y \notin \mathsf{Dom}(\Delta_2)$, $\Gamma_2(y) = \Gamma'_2(y)$.

  Therefore, with $\Gamma' = \Gamma'_1 \cap \Gamma'_2$, $\Delta = \Delta_1 \cap \Delta_2$ and $U = U_1 \cap U_2$:
  - $\Gamma = \Gamma_1 \cap \Gamma_2 \approx (\Gamma'_1 \cap \Delta_1) \cap (\Gamma'_2 \cap \Delta_2) \approx (\Gamma'_1 \cap \Gamma'_2) \cap (\Delta_1 \cap \Delta_2)$.
  - We have $\Gamma'_1 \cap \Gamma'_2, x : U_1 \cap U_2 \vdash_{\text{ns}} M : A_1 \cap A_2$ and, by Lemma 20.1, $\Delta_1 \cap \Delta_2 \vdash_{\text{ns}} N : U_1 \cap U_2$.
  - Assume $y \notin \mathsf{Dom}(\Delta_1 \cap \Delta_2)$. Then, $y \notin \mathsf{Dom}(\Delta_1)$ and $y \notin \mathsf{Dom}(\Delta_2)$. Therefore, $\Gamma(y) = (\Gamma_1 \cap \Gamma_2)(y) = \Gamma_1(y) \cap \Gamma_2(y) = \Gamma'_1(y) \cap \Gamma'_2(y) = (\Gamma'_1 \cap \Gamma'_2)(y)$.
- If $M = x$ and A is of the form F: Then, $M\{x := N\} = N$. Therefore, with $\Gamma' = ()$, $\Delta = \Gamma$ and $U = F$:
  - $\Gamma = () \cap \Gamma$.
  - We have $x : F \vdash_{\text{ns}} x : F$ and $\Gamma \vdash_{\text{ns}} N : F$.
  - Assume $y \notin \mathsf{Dom}(\Gamma)$. Therefore, $\Gamma(y) = \omega = ()(y)$.
- If M is of the form y with $y \neq x$ and A if of the form F: We have $M\{x := N\} = y = M$. Hence, $\Gamma = (y : F)$. Therefore, with $\Gamma' = (y : F)$, $\Delta = ()$ and $U = \omega$:
  - $(y : F) = (y : F) \cap ()$.
  - We have $y : F, x : \omega \vdash_{\text{ns}} M : F$ and, by the rule (ω), $\vdash_{\text{ns}} N : \omega$.
  - Assume $z \notin \mathsf{Dom}(())$. Therefore, $\Gamma(z) = (y : F)(z)$.
- If M is of the form $\lambda y.M_1$ with $y \neq x$ and $y \notin \text{fv}(N)$, and A is of the form $B \to F$. We have $(\lambda y.M_1)\{x := N\} = \lambda y.M_1\{x := N\}$. Therefore, there exists V such that $\Gamma, y : V \vdash_{\text{ns}} M_1\{x := N\} : F$ and $V = B$ or $V = \omega$ and output(B). By induction hypothesis, there exist Γ', Δ and U such that:
  - $(\Gamma, y : V) \approx \Gamma' \cap \Delta$.
  - We have $\Gamma', x : U \vdash_{\text{ns}} M_1 : F$ and $\Delta \vdash_{\text{ns}} N : U$.
  - For all $z \notin \mathsf{Dom}(\Delta)$, $(\Gamma, y : V)(z) = \Gamma'(z)$.

  There exist a unique Γ'' and a unique V' such that $(\Gamma'', y : V') = \Gamma'$. By Lemma 20.5, $\mathsf{Dom}(\Delta) \subseteq \text{fv}(N)$. Therefore, $y \notin \mathsf{Dom}(\Delta)$. Hence, $(\Gamma'', y : V') \cap \Delta = (\Gamma'' \cap \Delta, y : V')$. Therefore, $(\Gamma, y : V) \approx \Gamma' \cap \Delta = (\Gamma'' \cap \Delta, y : V')$. Hence, $\Gamma \approx \Gamma'' \cap \Delta$ and $V \approx V'$. By the fact that $y \notin \mathsf{Dom}(\Delta)$, we have $V = (\Gamma, y : V)(y) = \Gamma'(y) = (\Gamma'', y : V')(y) = V'$. Therefore:
  - We have $\Gamma \approx \Gamma'' \cap \Delta$.
  - We have $\Gamma'', y : V, x : U \vdash_{\text{ns}} M_1 : F$ and $\Delta \vdash_{\text{ns}} N : U$. Therefore, $\Gamma'', x : U \vdash_{\text{ns}} \lambda y.M_1 : B \to F$.

  - Assume $z \notin \mathsf{Dom}(\Delta)$. Then, $\Gamma(z) \approx \Gamma''(z) \cap \Delta(z) = \Gamma''(z) \cap \omega = \Gamma''(z)$.
    If $z \in \mathsf{Dom}(\Gamma)$: Then, $z \in \mathsf{Dom}(\Gamma'')$ and $\Gamma(z) = (\Gamma, y : V)(z) = \Gamma'(z) = (\Gamma'', y : V)(z) = \Gamma''(z)$.
    If $z \notin \mathsf{Dom}(\Gamma)$: Then, $z \notin \mathsf{Dom}(\Gamma'')$ and $\Gamma(z) = \Gamma''(z) = \omega$.
- If M is of the form $\lambda y.M_1$ with $y \neq x$ and $y \notin \mathrm{fv}(N)$, and A is of the form $[\lambda v]$. We have $(\lambda y.M_1)\{x := N\} = \lambda y.M_1\{x := N\}$. Therefore, there exist V and F such that $\Gamma, y : V \vdash_{\mathrm{ns}} M_1\{x := N\} : [v]$ and $\mathrm{input}(V)$. By induction hypothesis, there exist Γ', Δ and U such that:
  - $(\Gamma, y : V) \approx \Gamma' \cap \Delta$.
  - We have $\Gamma', x : U \vdash_{\mathrm{ns}} M_1 : [v]$ and $\Delta \vdash_{\mathrm{ns}} N : U$.
  - For all $z \notin \mathsf{Dom}(\Delta)$, $(\Gamma, y : V)(z) = \Gamma'(z)$.

  There exist a unique Γ'' and a unique V' such that $(\Gamma'', y : V') = \Gamma'$. By Lemma 20.5, $\mathsf{Dom}(\Delta) \subseteq \mathrm{fv}(N)$. Therefore, $y \notin \mathsf{Dom}(\Delta)$. Hence, $(\Gamma'', y : V') \cap \Delta = (\Gamma'' \cap \Delta, y : V')$. Therefore, $(\Gamma, y : V) \approx \Gamma' \cap \Delta = (\Gamma'' \cap \Delta, y : V')$. Hence, $\Gamma \approx \Gamma'' \cap \Delta$ and $V \approx V'$. By the fact that $y \notin \mathsf{Dom}(\Delta)$, we have $V = (\Gamma, y : V)(y) = \Gamma'(y) = (\Gamma'', y : V')(y) = V'$. Therefore:
  - We have $\Gamma \approx \Gamma'' \cap \Delta$.
  - We have $\Gamma'', y : V, x : U \vdash_{\mathrm{ns}} M_1 : [v]$ and $\Delta \vdash_{\mathrm{ns}} N : U$. Therefore, $\Gamma'', x : U \vdash_{\mathrm{ns}} \lambda y.M_1 : [\lambda v]$.
  - Assume $z \notin \mathsf{Dom}(\Delta)$. Then, $\Gamma(z) \approx \Gamma''(z) \cap \Delta(z) = \Gamma''(z) \cap \omega = \Gamma''(z)$.
    If $z \in \mathsf{Dom}(\Gamma)$: Then, $z \in \mathsf{Dom}(\Gamma'')$ and $\Gamma(z) = (\Gamma, y : V)(z) = \Gamma'(z) = (\Gamma'', y : V)(z) = \Gamma''(z)$.
    If $z \notin \mathsf{Dom}(\Gamma)$: Then, $z \notin \mathsf{Dom}(\Gamma'')$ and $\Gamma(z) = \Gamma''(z) = \omega$.
- If M is of the form M_1M_2 and A is of the form F: We have $(M_1M_2)\{x := N\} = M_1\{x := N\}M_2\{x := N\}$. We are in one of the following cases:
  - There exist Γ_1, Γ_2 and B such that $\Gamma = \Gamma_1 \cap \Gamma_2$, $\Gamma_1 \vdash_{\mathrm{ns}} M_1\{x := N\} : B \to F$ and $\Gamma_2 \vdash_{\mathrm{ns}} M_2\{x := N\} : B$. By induction hypothesis, there exist Γ'_1, Γ'_2, Δ_1, Δ_2, U_1 and U_2 such that:
    * $\Gamma_1 \approx \Gamma'_1 \cap \Delta_1$ and $\Gamma_2 \approx \Gamma'_2 \cap \Delta_2$.
    * We have $\Gamma'_1, x : U_1 \vdash_{\mathrm{ns}} M_1 : B \to F$, $\Gamma'_2, x : U_2 \vdash_{\mathrm{ns}} M_2 : B$, $\Delta_1 \vdash_{\mathrm{ns}} N : U_1$ and $\Delta_2 \vdash_{\mathrm{ns}} N : U_2$.
    * For all $y \notin \mathsf{Dom}(\Delta_1)$, $\Gamma_1(y) = \Gamma'_1(y)$.
    * For all $y \notin \mathsf{Dom}(\Delta_2)$, $\Gamma_2(y) = \Gamma'_2(y)$.

    Therefore, with $\Gamma' = \Gamma'_1 \cap \Gamma'_2$, $\Delta = \Delta_1 \cap \Delta_2$ and $U = U_1 \cap U_2$:
    * $\Gamma_1 \cap \Gamma_2 \approx (\Gamma'_1 \cap \Delta_1) \cap (\Gamma'_2 \cap \Delta_2) \approx (\Gamma'_1 \cap \Gamma'_2) \cap (\Delta_1 \cap \Delta_2)$.
    * We have $\Gamma'_1 \cap \Gamma'_2, x : U_1 \cap U_2 \vdash_{\mathrm{ns}} M_1M_2 : F$ and, by Lemma 20.1, $\Delta_1 \cap \Delta_2 \vdash_{\mathrm{ns}} N : U_1 \cap U_2$.
    * Assume $y \notin \mathsf{Dom}(\Delta_1 \cap \Delta_2)$. Then, $y \notin \mathsf{Dom}(\Delta_1)$ and $y \notin \mathsf{Dom}(\Delta_2)$. Therefore, $\Gamma(y) = (\Gamma_1 \cap \Gamma_2)(y) = \Gamma_1(y) \cap \Gamma_2(y) = \Gamma'_1(y) \cap \Gamma'_2(y) = (\Gamma'_1 \cap \Gamma'_2)(y)$.
  - There exist Γ_1, Γ_2, k and v such that $\Gamma = \Gamma_1 \cap \Gamma_2$, $F = [kv]$, $\Gamma_1 \vdash_{\mathrm{ns}} M_1\{x := N\} : [k]$ and $\Gamma_2 \vdash_{\mathrm{ns}} M_2\{x := N\} : [v]$. By induction hypothesis, there exist Γ'_1, Γ'_2, Δ_1, Δ_2, U_1 and U_2 such that:
    * $\Gamma_1 \approx \Gamma'_1 \cap \Delta_1$ and $\Gamma_2 \approx \Gamma'_2 \cap \Delta_2$.
    * We have $\Gamma'_1, x : U_1 \vdash_{\mathrm{ns}} M_1 : [k]$, $\Gamma'_2, x : U_2 \vdash_{\mathrm{ns}} M_2 : [v]$, $\Delta_1 \vdash_{\mathrm{ns}} N : U_1$ and $\Delta_2 \vdash_{\mathrm{ns}} N : U_2$.
    * For all $y \notin \mathsf{Dom}(\Delta_1)$, $\Gamma_1(y) = \Gamma'_1(y)$.
    * For all $y \notin \mathsf{Dom}(\Delta_2)$, $\Gamma_2(y) = \Gamma'_2(y)$.

    Therefore, with $\Gamma' = \Gamma'_1 \cap \Gamma'_2$, $\Delta = \Delta_1 \cap \Delta_2$ and $U = U_1 \cap U_2$:
    * $\Gamma_1 \cap \Gamma_2 \approx (\Gamma'_1 \cap \Delta_1) \cap (\Gamma'_2 \cap \Delta_2) \approx (\Gamma'_1 \cap \Gamma'_2) \cap (\Delta_1 \cap \Delta_2)$.
    * We have $\Gamma'_1 \cap \Gamma'_2, x : U_1 \cap U_2 \vdash_{\mathrm{ns}} M_1M_2 : [kv]$ and, by Lemma 20.1, $\Delta_1 \cap \Delta_2 \vdash_{\mathrm{ns}} N : U_1 \cap U_2$.
    * Assume $y \notin \mathsf{Dom}(\Delta_1 \cap \Delta_2)$. Then, $y \notin \mathsf{Dom}(\Delta_1)$ and $y \notin \mathsf{Dom}(\Delta_2)$. Therefore, $\Gamma(y) = (\Gamma_1 \cap \Gamma_2)(y) = \Gamma_1(y) \cap \Gamma_2(y) = \Gamma'_1(y) \cap \Gamma'_2(y) = (\Gamma'_1 \cap \Gamma'_2)(y)$.

◀

▶ **Lemma 26** (Typing accumulators).
If $\Gamma \vdash M : F$, input(Γ) and acc(M), then F is of the form $[k]$.

Proof. By induction on acc(M).

- For $\dfrac{}{\text{acc}(x)}$ with $M = x$: Then, $\Gamma = (x : F)$. Hence, input(F). Therefore, $F = [\triangledown]$ which is of the form $[k]$.
- For $\dfrac{\text{acc}(M_1)}{\text{acc}(M_1 M_2)}$ with $M = M_1 M_2$: Then, we are in one of the following cases:
  - There exist Γ_1, Γ_2 and A such that $\Gamma = \Gamma_1 \cap \Gamma_2$, $\Gamma_1 \vdash M_1 : A \to F$ and $\Gamma_2 \vdash M_2 : A$. By Lemma 17.10, we have input(Γ_1). By induction hypothesis, $A \to F$ is of the form $[k]$. Contradiction.
  - There exist Γ_1, Γ_2, k and v such that $\Gamma = \Gamma_1 \cap \Gamma_2$, $F = [kv]$, $\Gamma_1 \vdash M_1 : [k]$ and $\Gamma_2 \vdash M_2 : [v]$. Then, we can conclude.

◀

▶ **Lemma 27** (Typing normal forms).
If M cannot be reduced by $\longrightarrow_\beta$, then there exist Γ and v such that $\Gamma \vdash_{opt} M : [v]$.

Proof. By induction on M.

By Lemma 5, either M is of the form $\lambda x.M_1$ or we have acc(M). Hence, we are in one of the following cases:

- M is of the form $\lambda x.M_1$: By induction hypothesis, there exist Γ and v such that $\Gamma \vdash_{\text{opt}} M_1 : [v]$. Therefore, $\Gamma \vdash_{\text{ns}} M_1 : [v]$ and input(Γ). There exist a unique Γ_1 and a unique U such that $\Gamma = (\Gamma_1, x : U)$. Hence, input$(\Gamma_1)$ and input(U). Then, $\Gamma_1 \vdash_{\text{ns}} \lambda x.M_1 : [\lambda v]$ with input(Γ_1). Therefore, $\Gamma_1 \vdash_{\text{opt}} M : [\lambda v]$.
- M is a variable x: Then, $x : [\triangledown] \vdash_{\text{opt}} M : [\triangledown]$.
- M is of the form $M_1 M_2$ with acc(M_1): Then, M_1 and M_2 cannot be reduced by $\longrightarrow_\beta$. By induction hypothesis, there exist Γ_1, Γ_2, v_1 and v_2 such that $\Gamma_1 \vdash_{\text{opt}} M_1 : [v_1]$ and $\Gamma_2 \vdash_{\text{opt}} M_2 : [v_2]$. Therefore, $\Gamma_1 \vdash_{\text{ns}} M_1 : [v_1]$, $\Gamma_2 \vdash_{\text{ns}} M_2 : [v_2]$, input$(\Gamma_1)$ and input(Γ_2). By Lemma 26, v_1 if of the form k_1. By Lemma 17.11, we have input$(\Gamma_1 \cap \Gamma_2)$. Hence, $\Gamma_1 \cap \Gamma_2 \vdash_{\text{ns}} M_1 M_2 : [k_1 v_2]$ with input$(\Gamma_1 \cap \Gamma_2)$. Therefore, $\Gamma_1 \cap \Gamma_2 \vdash_{\text{opt}} M : [k_1 v_2]$.

◀

▶ **Theorem 28** (Subject Expansion).
If $\Gamma' \vdash_{opt} M' : F$ and $M \rightsquigarrow M'$, then there exists Γ such that $\Gamma \subseteq \Gamma'$ and $\Gamma \vdash_{opt} M : F$.

Proof. We prove by induction on $M \rightsquigarrow M'$ and $M \Rightarrow M'$ that if $\Gamma' \vdash_{\text{ns}} M : F$ and input(Γ'), and if we are in one of the following cases:

- We have $M \Rightarrow M'$
- We have $M \rightsquigarrow M'$ and output(F).

Then, there exists Γ such that $\Gamma \subseteq \Gamma'$, input(Γ), and $\Gamma \vdash_{\text{ns}} M : F$.

- For $\dfrac{x \in \text{fv}(M_1)}{(\lambda x.M_1)M_2 \Rightarrow M_1\{x := \ M_2\}}$ with $M' = M_1\{x := \ M_2\}$: By Lemma 25, there exist Γ_1, Γ_2 and U such that $\Gamma' \approx \Gamma_1 \cap \Gamma_2$, $\Gamma_1, x : U \vdash_{\text{ns}} M_1 : F$ and $\Gamma_2 \vdash_{\text{ns}} M_2 : U$. By Lemma 20.4, U is of the form A. Therefore, $\Gamma_1 \vdash_{\text{ns}} \lambda x.M_1 : A \to F$. Hence, $\Gamma_1 \cap \Gamma_2 \vdash_{\text{ns}} (\lambda x.M_1)M_2 : F$ with $\Gamma_1 \cap \Gamma_2 \subseteq \Gamma'$ and, by Lemma 17.10, input$(\Gamma_1 \cap \Gamma_2)$ (because $\Gamma' \subseteq \Gamma_1 \cap \Gamma_2$).
- For $\dfrac{x \notin \text{fv}(M_1) \quad M_2 \rightsquigarrow M_2'}{(\lambda x.M_1)M_2 \Rightarrow (\lambda x.M_1)M_2'}$ with $M' = (\lambda x.M_1)M_2'$: Then, we are in one of the following cases:

  - There exist Γ_1, Γ'_2 and A such that $\Gamma' = \Gamma_1 \cap \Gamma'_2$, $\Gamma_1 \vdash_{\text{ns}} \lambda x.M_1 : A \to F$ and $\Gamma'_2 \vdash_{\text{ns}} M'_2 : A$. Therefore, by the fact that there are no subsumptions, there exists U such that $\Gamma_1, x : U \vdash_{\text{ns}} M_1 : F$ and we have either $U = A$ or $U = \omega$ and $\text{output}(A)$. If $U = A$ then, by Lemma 20.4, $x \in \text{fv}(M_1)$: contradiction. Hence, $U = \omega$ and $\text{output}(A)$. Therefore, A is of the form G. By Lemma 17.10, we have $\text{input}(\Gamma_1)$ and $\text{input}(\Gamma'_2)$. By induction hypothesis, there exists Γ_2 such that $\text{input}(\Gamma_2)$, $\Gamma_2 \subseteq \Gamma'_2$ and $\Gamma_2 \vdash_{\text{ns}} M_2 : G$. By Lemma 17.11, $\text{input}(\Gamma_1 \cap \Gamma_2)$. Therefore, $\Gamma_1 \cap \Gamma_2 \vdash_{\text{ns}} (\lambda x.M_1)M_2 : F$ with $\Gamma_1 \cap \Gamma_2 \subseteq \Gamma_1 \cap \Gamma'_2 = \Gamma'$ and $\text{input}(\Gamma_1 \cap \Gamma_2)$.
  - There exist Γ_1, Γ_2, k and v such that $\Gamma' = \Gamma_1 \cap \Gamma_2$, $\Gamma_1 \vdash_{\text{ns}} \lambda x.M_1 : [k]$ and $\Gamma_2 \vdash_{\text{ns}} M'_2 : [v]$. An abstraction $\lambda x.M_1$ cannot have $[k]$ as a type (it is either an arrow $A \to G$ or of the form $[\lambda v_1]$). Contradiction.

- For $\dfrac{x \notin \text{fv}(M_1) \quad M_2 \text{ cannot be reduced by } \longrightarrow_\beta}{(\lambda x.M_1)M_2 \Rightarrow M_1}$ with $M' = M_1$: By Lemma 20.4, $x \notin \mathsf{Dom}(\Gamma')$. Hence, $\Gamma' = (\Gamma', x : \omega)$. By Lemma 27, there exist Γ_2 and G such that $\Gamma_2 \vdash_{\text{opt}} M_2 : G$. Hence, $\text{input}(\Gamma_2)$, $\text{output}(G)$ and $\Gamma_2 \vdash_{\text{ns}} M_2 : G$. Therefore, $\Gamma' \vdash_{\text{ns}} \lambda x.M_1 : G \to F$. By Lemma 17.11, we have $\text{input}(\Gamma' \cap \Gamma_2)$. Therefore, $\Gamma' \cap \Gamma_2 \vdash_{\text{ns}} (\lambda x.M_1)M_2 : F$ with $\Gamma' \cap \Gamma_2 \subseteq \Gamma'$ and $\text{input}(\Gamma' \cap \Gamma_2)$.
- For $\dfrac{M_1 \Rightarrow M'_1}{M_1M_2 \Rightarrow M'_1M_2}$ with $M' = M'_1M_2$, we are in one of the following cases:
  - There exist Γ'_1, Γ_2 and A such that $\Gamma' = \Gamma'_1 \cap \Gamma_2$, $\Gamma'_1 \vdash_{\text{ns}} M'_1 : A \to F$ and $\Gamma_2 \vdash_{\text{ns}} M_2 : A$: Then, by Lemma 17.10, $\text{input}(\Gamma'_1)$ and $\text{input}(\Gamma_2)$. By induction hypothesis, there exists Γ_1 such that $\Gamma_1 \subseteq \Gamma'_1$, $\text{input}(\Gamma_1)$ and $\Gamma_1 \vdash_{\text{ns}} M_1 : A \to F$. By Lemma 17.11, $\text{input}(\Gamma_1 \cap \Gamma_2)$. Therefore, $\Gamma_1 \cap \Gamma_2 \vdash_{\text{ns}} M_1M_2 : F$ with $\Gamma_1 \cap \Gamma_2 \subseteq \Gamma'_1 \cap \Gamma_2 = \Gamma'$ and $\text{input}(\Gamma_1 \cap \Gamma_2)$.
  - There exist Γ'_1, Γ_2, k and v such that $\Gamma' = \Gamma'_1 \cap \Gamma_2$, $F = [kv]$, $\Gamma'_1 \vdash_{\text{ns}} M'_1 : [k]$ and $\Gamma_2 \vdash_{\text{ns}} M_2 : [v]$: Then, by Lemma 17.10, $\text{input}(\Gamma'_1)$ and $\text{input}(\Gamma_2)$. By induction hypothesis, there exists Γ_1 such that $\Gamma_1 \subseteq \Gamma'_1$, $\text{input}(\Gamma_1)$ and $\Gamma_1 \vdash_{\text{ns}} M_1 : [k]$. By Lemma 17.11, $\text{input}(\Gamma_1 \cap \Gamma_2)$. Therefore, $\Gamma_1 \cap \Gamma_2 \vdash_{\text{ns}} M_1M_2 : [kv]$ with $\Gamma_1 \cap \Gamma_2 \subseteq \Gamma'_1 \cap \Gamma_2 = \Gamma'$ and $\text{input}(\Gamma_1 \cap \Gamma_2)$.
- For $\dfrac{\text{acc}(M_1) \quad M_2 \rightsquigarrow M'_2}{M_1M_2 \Rightarrow M_1M'_2}$ with $M' = M_1M'_2$, we are in one of the two following cases:
  - There exist Γ_1, Γ_2 and A such that $\Gamma' = \Gamma_1 \cap \Gamma_2$, $\Gamma_1 \vdash_{\text{ns}} M_1 : A \to F$ and $\Gamma_2 \vdash_{\text{ns}} M'_2 : A$. By Lemma 17.10, $\text{input}(\Gamma_1)$. By Lemma 26, $A \to F$ is of the form $[k]$. Contradiction.
  - There exist Γ_1, Γ'_2, k and v such that $\Gamma' = \Gamma_1 \cap \Gamma'_2$, $F = [kv]$, $\Gamma_1 \vdash_{\text{ns}} M_1 : [k]$ and $\Gamma'_2 \vdash_{\text{ns}} M'_2 : [v]$. By Lemma 17.10, $\text{input}(\Gamma_1)$ and $\text{input}(\Gamma'_2)$. By induction hypothesis, there exists Γ_2 such that $\Gamma_2 \subseteq \Gamma'_2$, $\text{input}(\Gamma_2)$ and $\Gamma_2 \vdash_{\text{ns}} M_2 : [v]$. By Lemma 17.11, $\text{input}(\Gamma_1 \cap \Gamma_2)$. Therefore, $\Gamma_1 \cap \Gamma_2 \vdash_{\text{ns}} M_1M_2 : [kv]$ with $\Gamma_1 \cap \Gamma_2 \subseteq \Gamma_1 \cap \Gamma'_2 = \Gamma'$ and $\text{input}(\Gamma_1 \cap \Gamma'_2)$.
- For $\dfrac{M \Rightarrow M'}{M \rightsquigarrow M'}$: Trivial.
- For $\dfrac{M_1 \rightsquigarrow M'_1}{\lambda x.M_1 \rightsquigarrow \lambda x.M'_1}$ with $M' = \lambda x.M'_1$ and $\text{output}(F)$: We have $\text{output}(F)$, so F is not an arrow $A \to G$. Therefore, there exist U' and v such that $F = [\lambda v]$, $\text{input}(U')$ and $\Gamma', x : U' \vdash_{\text{ns}} M'_1 : [v]$. Hence, $\text{input}(\Gamma', x : U')$. By induction hypothesis, there exists Γ_1 such that $\Gamma_1 \subseteq (\Gamma', x : U')$, $\text{input}(\Gamma_1)$ and $\Gamma_1 \vdash_{\text{ns}} M_1 : [v]$. There exist an unique Γ and a unique U such that $\Gamma_1 = (\Gamma, x : U)$. Therefore, $\Gamma \subseteq \Gamma'$, $U \subseteq U'$, $\text{input}(\Gamma)$ and $\text{input}(U)$. Hence, $\Gamma \vdash_{\text{ns}} \lambda x.M_1 : [\lambda v]$ with $\text{input}(\Gamma)$.

◀

▶ **Lemma 30** (Refined Substitution Lemma).
If $\Gamma, x : U \vdash^n_{ns} M : A$ and $\Delta \vdash^m_{ns} N : U$, then there exists Γ' such that:
- $\Gamma' \approx \Gamma \cap \Delta$.
- $\Gamma' \vdash^{n+m}_{ns} M\{x := N\} : A$.
- *For all $y \notin$ Dom(Δ), $\Gamma(y) = \Gamma'(y)$.*

Proof. By induction on $\Gamma, x : U \vdash_{\text{ns}} M : A$.

- For $\dfrac{}{x : F \vdash^0_{\text{ns}} x : F}$ with $\Gamma = ()$, $n = 0$, $U = F$, $M = x$ and $A = F$: We have $M\{x := N\} = N$. Therefore, with $\Gamma' = \Delta$:
  - $\Delta = () \cap \Delta$.
  - $\Delta \vdash^m_{\text{ns}} M\{x := N\} : F$ and $n + m = m$.
  - Assume $y \notin \mathsf{Dom}(\Delta)$. Then, $\Delta(y) = \omega = ()(y)$.
- For $\dfrac{}{y : F \vdash^0_{\text{ns}} y : F}$ with $x \neq y$, $\Gamma = (y : F)$, $n = 0$, $U = \omega$, $M = y$ and $A = F$: We have $M\{x := N\} = y = M$. By hypothesis, $\Delta \vdash^m_{\text{ns}} N : \omega$. Hence, $m = 0$ and $\Delta = ()$. Therefore, with $\Gamma' = \Gamma = (y : F)$:
  - $(y : F) = \Gamma = \Gamma \cap ()$.
  - $y : F \vdash^0_{\text{ns}} M\{x := N\} : F$ and $n + m = 0$.
  - Assume $z \notin \mathsf{Dom}(())$. Then, $(y : F)(z) = \Gamma(z)$.
- For $\dfrac{\Gamma_1, x : U_1 \vdash^{n_1}_{\text{ns}} M : A_1 \quad \Gamma_2, x : U_2 \vdash^{n_2}_{\text{ns}} M : A_2}{\Gamma_1 \cap \Gamma_2, x : U_1 \cap U_2 \vdash^{n_1+n_2}_{\text{ns}} M : A_1 \cap A_2}$ with $\Gamma = \Gamma_1 \cap \Gamma_2$, $n = n_1 + n_2$, $U = U_1 \cap U_2$ and $A = A_1 \cap A_2$. By hypothesis, $\Delta \vdash^m_{\text{ns}} N : U_1 \cap U_2$. By Lemma 20.1, there exist Δ_1, Δ_2, m_1 and m_2 such that $\Delta = \Delta_1 \cap \Delta_2$, $m = m_1 + m_2$, $\Delta_1 \vdash^{m_1}_{\text{ns}} N : U_1$ and $\Delta_2 \vdash^{m_2}_{\text{ns}} N : U_2$. By induction hypothesis, there exist Γ'_1 and Γ'_2 such that:
  - $\Gamma'_1 \approx \Gamma_1 \cap \Delta_1$ and $\Gamma'_2 \approx \Gamma_2 \cap \Delta_2$.
  - $\Gamma'_1 \vdash^{n_1+m_1}_{\text{ns}} M\{x := N\} : A_1$ and $\Gamma'_2 \vdash^{n_2+m_2}_{\text{ns}} M\{x := N\} : A_2$.
  - For all $y \notin \mathsf{Dom}(\Delta_1)$, $\Gamma_1(y) = \Gamma'_1(y)$.
  - For all $y \notin \mathsf{Dom}(\Delta_2)$, $\Gamma_2(y) = \Gamma'_2(y)$.

  Therefore, with $\Gamma' = \Gamma'_1 \cap \Gamma'_2$:
  - $\Gamma'_1 \cap \Gamma'_2 \approx (\Gamma_1 \cap \Delta_1) \cap (\Gamma_2 \cap \Delta_2) \approx (\Gamma_1 \cap \Gamma_2) \cap (\Delta_1 \cap \Delta_2) = \Gamma \cap \Delta$.
  - We have $\Gamma'_1 \cap \Gamma'_2 \vdash^{n_1+m_1+n_2+m_2}_{\text{ns}} M\{x := N\} : A_1 \cap A_2$ and $n + m = n_1 + m_1 + n_2 + m_2$.
  - Assume $y \notin \mathsf{Dom}(\Delta) = \mathsf{Dom}(\Delta_1 \cap \Delta_2)$. Then, $y \notin \mathsf{Dom}(\Delta_1)$ and $y \notin \mathsf{Dom}(\Delta_2)$. Therefore, $\Gamma(y) = (\Gamma_1 \cap \Gamma_2)(y) = \Gamma_1(y) \cap \Gamma_2(y) = \Gamma'_1(y) \cap \Gamma'_2(y) = (\Gamma'_1 \cap \Gamma'_2)(y)$.
- For $\dfrac{\Gamma, x : U, y : V \vdash^n_{\text{ns}} M_1 : F}{\Gamma, x : U \vdash^n_{\text{ns}} \lambda y.M_1 : B \to F}$ with $M = \lambda y.M_1$, $y \notin \text{fv}(N)$, $y \neq x$, $A = B \to F$ and $V = B$ or $V = \omega$ and $\text{output}(B)$. We have $(\lambda y.M_1)\{x := N\} = \lambda y.M_1\{x := N\}$. By induction hypothesis, there exist Γ' such that:
  - $\Gamma' \approx (\Gamma, y : V) \cap \Delta$.
  - $\Gamma' \vdash^{n+m}_{\text{ns}} M_1\{x := N\} : F$.
  - For all $z \notin \mathsf{Dom}(\Delta)$, $(\Gamma, y : V)(z) = \Gamma'(z)$.

  By Lemma 20.5, $\mathsf{Dom}(\Delta) \subseteq \text{fv}(N)$. Therefore, $y \notin \mathsf{Dom}(\Delta)$ and $(\Gamma, y : V) \cap \Delta = (\Gamma \cap \Delta, y : V)$. There exist a unique Γ'' and a unique V' such that $\Gamma' = (\Gamma'', y : V')$. Hence, $(\Gamma'', y : V') \approx (\Gamma, y : V) \cap \Delta = (\Gamma \cap \Delta, y : V)$. Therefore, $\Gamma'' \approx \Gamma \cap \Delta$ and $V' \approx V$. We have $y \notin \mathsf{Dom}(\Delta)$, so $V = (\Gamma, y : V)(y) = \Gamma'(y) = (\Gamma'', y : V')(y) = V'$. Therefore:
  - $\Gamma'' \approx \Gamma \cap \Delta$.
  - We have $\Gamma'', y : V \vdash^{n+m}_{\text{ns}} M_1\{x := N\} : F$. Therefore, $\Gamma'' \vdash^{n+m}_{\text{ns}} \lambda y.M_1\{x := N\} : B \to F$.
  - Assume $z \notin \mathsf{Dom}(\Delta)$. Then, $\Gamma''(z) \approx (\Gamma \cap \Delta)(z) = \Gamma(z) \cap \Delta(z) = \Gamma(z) \cap \omega = \Gamma(z)$.
    If $z \in \mathsf{Dom}(\Gamma)$: Then, $z \in \mathsf{Dom}(\Gamma'')$ and $\Gamma(y) = (\Gamma, y : V)(z) = \Gamma'(z) = (\Gamma'', y : V)(z) = \Gamma''(z)$.
    If $z \notin \mathsf{Dom}(\Gamma)$: Then, $z \notin \mathsf{Dom}(\Gamma'')$ and $\Gamma(z) = \Gamma''(z) = \omega$.
- For $\dfrac{\Gamma, x : U, y : V \vdash^n_{\text{ns}} M_1 : [v] \quad \text{input}(V)}{\Gamma, x : U \vdash^n_{\text{ns}} \lambda y.M_1 : [\lambda v]}$ with $M = \lambda y.M_1$, $y \notin \text{fv}(N)$, $y \neq x$ and $A = [\lambda v]$. We have $(\lambda y.M_1)\{x := N\} = \lambda y.M_1\{x := N\}$. By induction hypothesis, there exist Γ' such that:
  - $\Gamma' \approx (\Gamma, y : V) \cap \Delta$.
  - $\Gamma' \vdash^{n+m}_{\text{ns}} M_1\{x := N\} : [v]$.
  - For all $z \notin \mathsf{Dom}(\Delta)$, $(\Gamma, y : V)(z) = \Gamma'(z)$.

By Lemma 20.5, $\mathsf{Dom}(\Delta) \subseteq \mathrm{fv}(N)$. Therefore, $y \notin \mathsf{Dom}(\Delta)$ and $(\Gamma, y : V) \cap \Delta = (\Gamma \cap \Delta, y : V)$. There exist a unique Γ'' and a unique V' such that $\Gamma' = (\Gamma'', y : V')$. Hence, $(\Gamma'', y : V') \approx (\Gamma, y : V) \cap \Delta = (\Gamma \cap \Delta, y : V)$. Therefore, $\Gamma'' \approx \Gamma \cap \Delta$ and $V' \approx V$. We have $y \notin \mathsf{Dom}(\Delta)$, so $V = (\Gamma, y : V)(y) = \Gamma'(y) = (\Gamma'', y : V')(y) = V'$. Therefore:

- $\Gamma'' \approx \Gamma \cap \Delta$.
- We have $\Gamma'', y : V \vdash^{n+m}_{\mathrm{ns}} M_1\{x := N\} : [v]$. Therefore, $\Gamma'' \vdash^{n+m}_{\mathrm{ns}} \lambda y.M_1\{x := N\} : [\lambda v]$.
- Assume $z \notin \mathsf{Dom}(\Delta)$. Then, $\Gamma''(z) \approx (\Gamma \cap \Delta)(z) = \Gamma(z) \cap \Delta(z) = \Gamma(z) \cap \omega = \Gamma(z)$.
  If $z \in \mathsf{Dom}(\Gamma)$: Then, $z \in \mathsf{Dom}(\Gamma'')$ and $\Gamma(y) = (\Gamma, y : V)(z) = \Gamma'(z) = (\Gamma'', y : V)(z) = \Gamma''(z)$.
  If $z \notin \mathsf{Dom}(\Gamma)$: Then, $z \notin \mathsf{Dom}(\Gamma'')$ and $\Gamma(z) = \Gamma''(z) = \omega$.

- For $\dfrac{\Gamma_1, x : U_1 \vdash^{n_1}_{\mathrm{ns}} M_1 : B \to F \quad \Gamma_2, x : U_2 \vdash^{n_2}_{\mathrm{ns}} M_2 : B}{\Gamma_1 \cap \Gamma_2, x : U_1 \cap U_2 \vdash^{n_1+n_2+1}_{\mathrm{ns}} M_1 M_2 : F}$ with $\Gamma = \Gamma_1 \cap \Gamma_2$, $n = n_1 + n_2 + 1$, $U = U_1 \cap U_2$, $M = M_1 M_2$ and $A = F$. We have $(M_1 M_2)\{x := N\} = M_1\{x := N\} M_2\{x := N\}$. By hypothesis, $\Delta \vdash^m_{\mathrm{ns}} N : U_1 \cap U_2$. By Lemma 20.1, there exist Δ_1, Δ_2, m_1 and m_2 such that $\Delta = \Delta_1 \cap \Delta_2$, $m = m_1 + m_2$, $\Delta_1 \vdash^{m_1}_{\mathrm{ns}} N : U_1$ and $\Delta_2 \vdash^{m_2}_{\mathrm{ns}} N : U_2$. By induction hypothesis, there exist Γ'_1 and Γ'_2 such that:
  - $\Gamma'_1 \approx \Gamma_1 \cap \Delta_1$ and $\Gamma'_2 \approx \Gamma_2 \cap \Delta_2$.
  - $\Gamma'_1 \vdash^{n_1+m_1}_{\mathrm{ns}} M_1 : B \to F$ and $\Gamma'_2 \vdash^{n_2+m_2}_{\mathrm{ns}} M_2 : B$.
  - For all $y \notin \mathsf{Dom}(\Delta_1)$, $\Gamma_1(y) = \Gamma'_1(y)$.
  - For all $y \notin \mathsf{Dom}(\Delta_2)$, $\Gamma_2(y) = \Gamma'_2(y)$.

  Therefore, with $\Gamma' = \Gamma'_1 \cap \Gamma'_2$:
  - $\Gamma'_1 \cap \Gamma'_2 \approx (\Gamma_1 \cap \Delta_1) \cap (\Gamma_2 \cap \Delta_2) \approx (\Gamma_1 \cap \Gamma_2) \cap (\Delta_1 \cap \Delta_2) = \Gamma \cap \Delta$.
  - We have $\Gamma'_1 \cap \Gamma'_2 \vdash^{n_1+m_1+n_2+m_2+1}_{\mathrm{ns}} M_1\{x := N\} M_2\{x := N\} : F$ and $n + m = n_1 + m_1 + n_2 + m_2 + 1$.
  - Assume $y \notin \mathsf{Dom}(\Delta) = \mathsf{Dom}(\Delta_1 \cap \Delta_2)$. Then, $y \notin \mathsf{Dom}(\Delta_1)$ and $y \notin \mathsf{Dom}(\Delta_2)$. Therefore, $\Gamma(y) = (\Gamma_1 \cap \Gamma_2)(y) = \Gamma_1(y) \cap \Gamma_2(y) = \Gamma'_1(y) \cap \Gamma'_2(y) = (\Gamma'_1 \cap \Gamma'_2)(y)$.

- For $\dfrac{\Gamma_1, x : U_1 \vdash^{n_1}_{\mathrm{ns}} M_1 : [k] \quad \Gamma_2, x : U_2 \vdash^{n_2}_{\mathrm{ns}} M_2 : [v]}{\Gamma_1 \cap \Gamma_2, x : U_1 \cap U_2 \vdash^{n_1+n_2}_{\mathrm{ns}} M_1 M_2 : [kv]}$ with $\Gamma = \Gamma_1 \cap \Gamma_2$, $n = n_1 + n_2$, $U = U_1 \cap U_2$, $M = M_1 M_2$ and $A = [kv]$. We have $(M_1 M_2)\{x := N\} = M_1\{x := N\} M_2\{x := N\}$. By hypothesis, $\Delta \vdash^m_{\mathrm{ns}} N : U_1 \cap U_2$. By Lemma 20.1, there exist Δ_1, Δ_2, m_1 and m_2 such that $\Delta = \Delta_1 \cap \Delta_2$, $m = m_1 + m_2$, $\Delta_1 \vdash^{m_1}_{\mathrm{ns}} N : U_1$ and $\Delta_2 \vdash^{m_2}_{\mathrm{ns}} N : U_2$. By induction hypothesis, there exist Γ'_1 and Γ'_2 such that:
  - $\Gamma'_1 \approx \Gamma_1 \cap \Delta_1$ and $\Gamma'_2 \approx \Gamma_2 \cap \Delta_2$.
  - $\Gamma'_1 \vdash^{n_1+m_1}_{\mathrm{ns}} M_1 : [k]$ and $\Gamma'_2 \vdash^{n_2+m_2}_{\mathrm{ns}} M_2 : [v]$.
  - For all $y \notin \mathsf{Dom}(\Delta_1)$, $\Gamma_1(y) = \Gamma'_1(y)$.
  - For all $y \notin \mathsf{Dom}(\Delta_2)$, $\Gamma_2(y) = \Gamma'_2(y)$.

  Therefore, with $\Gamma' = \Gamma'_1 \cap \Gamma'_2$:
  - $\Gamma'_1 \cap \Gamma'_2 \approx (\Gamma_1 \cap \Delta_1) \cap (\Gamma_2 \cap \Delta_2) \approx (\Gamma_1 \cap \Gamma_2) \cap (\Delta_1 \cap \Delta_2) = \Gamma \cap \Delta$.
  - We have $\Gamma'_1 \cap \Gamma'_2 \vdash^{n_1+m_1+n_2+m_2}_{\mathrm{ns}} M_1\{x := N\} M_2\{x := N\} : [kv]$ and $n + m = n_1 + m_1 + n_2 + m_2$.
  - Assume $y \notin \mathsf{Dom}(\Delta) = \mathsf{Dom}(\Delta_1 \cap \Delta_2)$. Then, $y \notin \mathsf{Dom}(\Delta_1)$ and $y \notin \mathsf{Dom}(\Delta_2)$. Therefore, $\Gamma(y) = (\Gamma_1 \cap \Gamma_2)(y) = \Gamma_1(y) \cap \Gamma_2(y) = \Gamma'_1(y) \cap \Gamma'_2(y) = (\Gamma'_1 \cap \Gamma'_2)(y)$.

◀

▶ **Lemma 31** (Measure of normal forms).
If $\Gamma \vdash^n M : F$, M cannot be reduced by $\longrightarrow_\beta$, input(Γ) and output(F), then $n = 0$.

Proof. By induction on M: By Lemma 5, M is of the form $\lambda x.M_1$ or $\mathrm{acc}(M)$. Therefore, we are in one of the following cases:

- M is of the form $\lambda x.M_1$: We have output(F), so F is not an arrow $A \to G$. Therefore, there exist U and G such that input(U), output(G) and $\Gamma, x : U \vdash^n M_1 : G$. Hence, input$(\Gamma, x : U)$. By induction hypothesis, $n = 0$.
- M is a variable x: Then, $n = 0$.

- M is of the form $M_1 M_2$ with $\mathrm{acc}(M_1)$: Then, we are in one of the following cases:
  - There exist Γ_1, Γ_2 and A such that $\Gamma = \Gamma_1 \cap \Gamma_2$, $\Gamma_1 \vdash M_1 : A \to F$ and $\Gamma_2 \vdash M_2 : A$. By Lemma 17.10, we have $\mathrm{input}(\Gamma_1)$. By Lemma 26, $A \to F$ is of the form $[k]$. Contradiction.
  - There exist Γ_1, Γ_2, n_1, n_2, k and v such that $\Gamma = \Gamma_1 \cap \Gamma_2$, $n = n_1 + n_2$, $F = [kv]$, $\Gamma_1 \vdash^{n_1} M_1 : [k]$ and $\Gamma_2 \vdash^{n_2} M_2 : [v]$. By Lemma 17.10, $\mathrm{input}(\Gamma_1)$ and $\mathrm{input}(\Gamma_2)$. By induction hypothesis, $n_1 = 0$ and $n_2 = 0$. Therefore $n = 0$.

◀

▶ **Theorem 32** (Refined Subject Reduction).
If $\Gamma \vdash^n_{opt} M : F$ and $M \rightsquigarrow M'$, then there exists Γ' such that $\Gamma \subseteq \Gamma'$ and $\Gamma' \vdash^{n-1}_{opt} M' : F$.

Proof. We prove by induction on $M \rightsquigarrow M'$ and $M \Rightarrow M'$ that if $\Gamma \vdash^n_{\mathrm{ns}} M : F$, $\mathrm{input}(\Gamma)$, and if we are in one the following cases:
- We have $M \rightsquigarrow M'$ and $\mathrm{output}(F)$.
- We have $M \Rightarrow M'$.

Then, there exists Γ' such that $\Gamma \subseteq \Gamma'$, $\Gamma' \vdash^{n-1}_{\mathrm{ns}} M' : F$ and then, by Lemma 17.10, we have $\mathrm{input}(\Gamma')$.

- For $\dfrac{x \in \mathrm{fv}(M_1)}{(\lambda x.M_1)M_2 \Rightarrow M_1\{x := \ M_2\}}$ with $M = (\lambda x.M_1)M_2$, we are in one of the following cases:
  - There exist Γ_1, Γ_2, n_1, n_2 and A such that $\Gamma = \Gamma_1 \cap \Gamma_2$, $n = n_1 + n_2 + 1$, $\Gamma_1 \vdash^{n_1}_{\mathrm{ns}} \lambda x.M_1 : A \to F$ and $\Gamma_2 \vdash^{n_2}_{\mathrm{ns}} M_2 : A$: By the fact that there are no subsumptions, there exists U such that $\Gamma_1, x : U \vdash^{n_1}_{\mathrm{ns}} M_1 : F$ and $U = A$ or $U = \omega$ and $\mathrm{output}(A)$. By Lemma 20.4, if $U = \omega$, then $x \notin \mathrm{fv}(M_1)$: contradiction. Therefore, $A = U$. By Lemma 30, there exist Γ' such that $\Gamma' \approx \Gamma_1 \cap \Gamma_2$ and $\Gamma' \vdash^{n_1+n_2}_{\mathrm{ns}} M_1\{x := \ M_2\} : F$ with $\Gamma \subseteq \Gamma'$.
  - There exist Γ_1, Γ_2, k and v such that $\Gamma = \Gamma_1 \cap \Gamma_2$, $F = [kv]$, $\Gamma_1 \vdash_{\mathrm{ns}} \lambda x.M_1 : [k]$ and $\Gamma_2 \vdash_{\mathrm{ns}} M_2 : [v]$. An abstraction $\lambda x.M_1$ cannot have $[k]$ as a type (it is either an arrow $A \to G$ or of the form $[\lambda v_1]$). Contradiction.
- For $\dfrac{x \notin \mathrm{fv}(M_1) \quad M_2 \rightsquigarrow M_2'}{(\lambda x.M_1)M_2 \Rightarrow (\lambda x.M_1)M_2'}$ with $M = (\lambda x.M_1)M_2$, we are in one of the following cases:
  - There exist Γ_1, Γ_2, n_1, n_2 and A such that $\Gamma = \Gamma_1 \cap \Gamma_2$, $n = n_1 + n_2 + 1$, $\Gamma_1 \vdash^{n_1}_{\mathrm{ns}} \lambda x.M_1 : A \to F$ and $\Gamma_2 \vdash^{n_2}_{\mathrm{ns}} M_2 : A$: By the fact that there are no subsumptions, there exists U such that $\Gamma_1, x : U \vdash^{n_1}_{\mathrm{ns}} M_1 : F$ and $U = A$ or $U = \omega$ and $\mathrm{output}(A)$. By Lemma 20.4, if $U = A$, then $x \in \mathrm{fv}(M_1)$: contradiction. Therefore, $U = \omega$ and $\mathrm{output}(A)$. Hence, A is of the form G. By Lemma 17.10, $\mathrm{input}(\Gamma_1)$ and $\mathrm{input}(\Gamma_2)$. By induction hypothesis, there exists Γ_2' such that $\Gamma_2 \subseteq \Gamma_2'$ and $\Gamma_2' \vdash^{n_2-1}_{\mathrm{ns}} M_2' : G$. Therefore, $\Gamma_1 \cap \Gamma_2' \vdash^{n_1+n_2}_{\mathrm{ns}} (\lambda x.M_1)M_2' : F$ with $\Gamma = \Gamma_1 \cap \Gamma_2 \subseteq \Gamma_1 \cap \Gamma_2'$.
  - There exist Γ_1, Γ_2, k and v such that $\Gamma = \Gamma_1 \cap \Gamma_2$, $F = [kv]$, $\Gamma_1 \vdash_{\mathrm{ns}} \lambda x.M_1 : [k]$ and $\Gamma_2 \vdash_{\mathrm{ns}} M_2 : [v]$. An abstraction $\lambda x.M_1$ cannot have $[k]$ as a type (it is either an arrow $A \to G$ or of the form $[\lambda v_1]$). Contradiction.
- For $\dfrac{x \notin \mathrm{fv}(M_1) \quad M_2 \text{ cannot be reduced by } \longrightarrow_\beta}{(\lambda x.M_1)M_2 \Rightarrow M_1}$ with $M = (\lambda x.M_1)M_2$, we are in one of the following cases:
  - There exist Γ_1, Γ_2, n_1, n_2 and A such that $\Gamma = \Gamma_1 \cap \Gamma_2$, $n = n_1 + n_2 + 1$, $\Gamma_1 \vdash^{n_1}_{\mathrm{ns}} \lambda x.M_1 : A \to F$ and $\Gamma_2 \vdash^{n_2}_{\mathrm{ns}} M_2 : A$: By the fact that there are no subsumptions, there exists U such that $\Gamma_1, x : U \vdash^{n_1} M_1 : F$ and $U = A$ or $U = \omega$ and $\mathrm{output}(A)$. By Lemma 20.4, if $U = A$, then $x \in \mathrm{fv}(M_1)$: contradiction. Therefore, $U = \omega$ and $\mathrm{output}(A)$. Hence, A is of the form G and $\Gamma_1 = (\Gamma_1, x : U)$. By Lemma 17.10, $\mathrm{input}(\Gamma_1)$ and $\mathrm{input}(\Gamma_2)$. By Lemma 31, $n_2 = 0$. Therefore, $\Gamma_1 \vdash^{n_1+n_2}_{\mathrm{ns}} M_1 : F$ with $\Gamma = \Gamma_1 \cap \Gamma_2 \subseteq \Gamma_1$.

  - There exist Γ_1, Γ_2, k and v such that $\Gamma = \Gamma_1 \cap \Gamma_2$, $F = [kv]$, $\Gamma_1 \vdash_{\mathrm{ns}} \lambda x.M_1 : [k]$ and $\Gamma_2 \vdash_{\mathrm{ns}} M_2 : [v]$. An abstraction $\lambda x.M_1$ cannot have $[k]$ as a type (it is either an arrow $A \to G$ or of the form $[\lambda v_1]$). Contradiction.
- For $\dfrac{M_1 \Rightarrow M_1'}{M_1 M_2 \Rightarrow M_1' M_2}$ with $M = M_1 M_2$, we are in one of the following cases:
  - There exist Γ_1, Γ_2, n_1, n_2 and A such that $\Gamma = \Gamma_1 \cap \Gamma_2$, $n = n_1 + n_2 + 1$, $\Gamma_1 \vdash^{n_1}_{\mathrm{ns}} M_1 : A \to F$ and $\Gamma_2 \vdash^{n_2}_{\mathrm{ns}} M_2 : A$: By Lemma 17.10, input$(\Gamma_1)$ and input(Γ_2). By induction hypothesis, there exist Γ_1' such that $\Gamma_1 \subseteq \Gamma_1'$ and $\Gamma_1' \vdash^{n_1 - 1} M_1' : A \to F$. Therefore, $\Gamma_1' \cap \Gamma_2 \vdash^{n_1+n_2}_{\mathrm{ns}} M_1' M_2 : F$ with $\Gamma = \Gamma_1 \cap \Gamma_2 \subseteq \Gamma_1' \cap \Gamma_2$.
  - There exist Γ_1, Γ_2, n_1, n_2, k and v such that $\Gamma = \Gamma_1 \cap \Gamma_2$, $n = n_1 + n_2$, $F = [kv]$, $\Gamma_1 \vdash^{n_1}_{\mathrm{ns}} M_1 : [k]$ and $\Gamma_2 \vdash^{n_2}_{\mathrm{ns}} M_2 : [v]$: By Lemma 17.10, input$(\Gamma_1)$ and input(Γ_2). By induction hypothesis, there exists Γ_1' such that $\Gamma_1 \subseteq \Gamma_1'$ and $\Gamma_1' \vdash^{n_1 - 1}_{\mathrm{ns}} M_1' : [k]$. Therefore, $\Gamma_1' \cap \Gamma_2 \vdash^{n_1+n_2-1}_{\mathrm{ns}} M_1' M_2 : [kv]$ with $\Gamma = \Gamma_1 \cap \Gamma_2 \subseteq \Gamma_1' \cap \Gamma_2$.
- For $\dfrac{\mathrm{acc}(M_1) \quad M_2 \rightsquigarrow M_2'}{M_1 M_2 \Rightarrow M_1 M_2'}$, we are in one of the following cases:
  - There exist Γ_1, Γ_2 and A such that $\Gamma = \Gamma_1 \cap \Gamma_2$, $\Gamma_1 \vdash_{\mathrm{ns}} M_1 : A \to F$ and $\Gamma_2 \vdash_{\mathrm{ns}} M_2 : A$. By Lemma 17.10, input$(\Gamma_1)$ and input(Γ_2). By Lemma 26, $A \to F$ is of the form $[k]$. Contradiction.
  - There exist Γ_1, Γ_2, n_1, n_2, k and v such that $\Gamma = \Gamma_1 \cap \Gamma_2$, $n = n_1 + n_2$, $F = [kv]$, $\Gamma_1 \vdash^{n_1}_{\mathrm{ns}} M_1 : [k]$ and $\Gamma_2 \vdash^{n_2}_{\mathrm{ns}} M_2 : [v]$. By Lemma 17.10, input$(\Gamma_1)$ and input(Γ_2). By induction hypothesis, there exists Γ_2' such that $\Gamma_2 \subseteq \Gamma_2'$ and $\Gamma_2' \vdash^{n_2 - 1} M_2' : [v]$. Therefore, $\Gamma_1 \cap \Gamma_2' \vdash^{n_1+n_2-1} M_1 M_2' : [kv]$ with $\Gamma = \Gamma_1 \cap \Gamma_2 \subseteq \Gamma_1 \cap \Gamma_2'$.
- For $\dfrac{M \Rightarrow M'}{M \rightsquigarrow M'}$: Trivial.
- For $\dfrac{M_1 \rightsquigarrow M_1'}{\lambda x.M_1 \rightsquigarrow \lambda x.M_1'}$ with $M = \lambda x.M_1$ and output(F): By the fact that we have output(F), F is not an arrow $A \to G$. Therefore, there exist U and v such that input(U), $F = [\lambda v]$ and $\Gamma, x : U \vdash^{n}_{\mathrm{ns}} M_1 : [v]$. Hence, input$(\Gamma, x : U)$. By induction hypothesis, there exists Γ_1 such that $(\Gamma, x : U) \subseteq \Gamma_1$ and $\Gamma_1 \vdash^{n-1}_{\mathrm{ns}} M_1' : [v]$. There exist an unique Γ' and a unique U' such that $\Gamma_1 = (\Gamma', x : U')$. Therefore, $\Gamma \subseteq \Gamma'$ and $U \subseteq U'$. By Lemma 14.5, we have input(U'). Hence, $\Gamma' \vdash^{n-1} \lambda x.M_1' : [\lambda v]$.

◀