JFP, **??** pages, 2024.
doi:10.1017/xxxxx

Local Type Inference for Polarised System F with Existentials

ILYA KAYSIN

University of Cambridge
(e-mail: ik404@cam.ac.uk)

NEEL KRISHNASWAMI

University of Cambridge
(e-mail: nk480@cl.cam.ac.uk)

Abstract

We study type inference for the calculus $\mathrm{F}^{\pm}_{\exists}$, which is a version of call-by-push-value extended with support for fully impredicative existential and universal quantifiers. We give a local type inference algorithm for this calculus as well as a declarative type system acting as a specification for the algorithm and show that the algorithm is sound and complete with respect to the declarative specification. The inclusion of existential quantifiers is unusual and supporting it required us to use a number of novel techniques including the combination of unification and anti-unification as part of the inference algorithm.

1 Introduction

Over the last half-century, there has been considerable work on developing type inference algorithms for programming languages, mostly aimed at solving the problem of *type polymorphism*.

That is, in pure polymorphic lambda calculus—System F (Girard, 1971; Reynolds, 1974)—the polymorphic type $\forall\alpha.\ A$ has a big lambda $\Lambda\alpha.\ e$ as an introduction form, and an explicit type application $e\ \{A\}$ as an elimination form. This is an extremely simple and compact type system, whose rules fit on a single page, but whose semantics are advanced enough to accurately represent concepts such as parametricity and representation independence. However, System F by itself is unwieldy as a programming language. The fact that the universal type $\forall\alpha.\ A$ has explicit eliminations means that programs written using polymorphic types will need to be stuffed to the gills with type annotations explaining how and when to instantiate the quantifiers.

Therefore, most work on type inference has been aimed at handling type instantiations implicitly—we want to be able to use a polymorphic function like $\texttt{len} : \forall\alpha.\ [\alpha] \to \texttt{Int}$ at

any concrete list type without explicitly instantiating the quantifier in `len`'s type. That is, we want to write `len` $[1, 2, 3]$ instead of writing `len` $\{\texttt{Int}\}$ $[1, 2, 3]$.

The way this is typically done is by using a *subtyping* relation, induced by the polymorphic instantiation: the type $\forall\alpha.\ A$ is a *subtype* of all of its instantiations. So we wish to be free to use the same polymorphic function `len` at many different types such as $[\texttt{Int}] \to \texttt{Int}$, $[\texttt{Bool}] \to \texttt{Int}$, $[\texttt{Int}] \times \texttt{Bool} \to \texttt{Int}$, and so on. However, the subtyping can be used nondeterministically: whenever we see a polymorphic type $\forall\alpha.\ A$, we know it is a subtype of *any* of its instantiations. To turn this into an algorithm, we have to actually make some choices.

The most famous of the algorithms for handling this is the Damas-Hindley-Milner (DHM) algorithm (Hindley, 1969; Milner, 1978; Damas and Milner, 1982). DHM deals with this choice using *unification*. Whenever we would have had to guess a particular concrete type in the specification, the DHM algorithm introduces a *unification variable*, and incrementally instantiates this variable as more and more type constraints are observed.

However, the universal quantifier is not the only quantifier! Dual to the universal quantifier $\forall$ is the existential quantifier $\exists$. Even though existential types have an equal logical status to universal types, they have been studied much less frequently in the literature. The most widely-used algorithm for handling existentials is the algorithm of Odersky and Läufer (Läufer and Odersky, 1994). This algorithm treats existentials in a second-class fashion: they are not explicit connectives of the logic but rather are tied to datatype declarations. As a result, packing and unpacking existential types is tied to constructor introductions and constructor eliminations in pattern matches. This allows Damas-Milner inference to continue to work almost unchanged, but it does come at the cost of losing first-class existentials and also of restricting the witness types to monomorphic types (i.e., types with no quantifiers inside of them).

There has been a limited amount of work on support for existential types as a first-class concept. In an unpublished draft, Leijen (2006) studied an extension of Damas-Milner inference in which type schemes contain alternating universal and existential quantifiers. Quantifiers still range over monotypes, so higher-rank polymorphism is not permitted: one cannot instantiate a quantifier with a type that itself contains quantifiers. More recently, Dunfield and Krishnaswami (2016) studied type inference for existential types in the context of GADT type inference, which, while still predicative, supported higher-rank types (i.e., quantifiers can occur anywhere inside of a type scheme). Eisenberg et al. (2021) propose a system which only permits types of the form forall-exists, but which permits projective elimination in the style of ML modules.

All of these papers are restricted to *predicative* quantification, where quantifiers can only be instantiated with monotypes. However, existential types in full System F are *impredicative*—that is, quantifiers can be instantiated with arbitrary types, specifically including types containing quantifiers.

Historically, inference for impredicative quantification has been neglected, due to results by Tiuryn and Urzyczyn (1996) and Chrzaszcz (1998) which show that not only is full type inference for System F undecidable, but even that the subtyping relation induced by type

instantiation is undecidable. However, in recent years, interest in impredicative inference has revived (for example, the work of Serrano et al. (2020)), with a focus on avoiding the undecidability barriers by doing *partial* type inference. That is, we no longer try to infer the type of any term in the language, but rather accept that the programmer will need to write annotations in cases where inference would be too difficult. Unfortunately, it is often difficult to give a specification for when the partial algorithm succeeds—for example, Eisenberg et al. (2021) observe that their algorithm lacks a declarative specification, and explain why supporting existential types makes giving a specification particularly difficult.

One especially well-behaved form of partial type inference is *local type inference*, introduced by Pierce and Turner (2000). In this approach, quantifiers in a polymorphic function are instantiated using only the type information available in the arguments at each call site. While this infers fewer quantifiers than global algorithms such as Damas-Milner can, it has the twin benefits that it is easy to give a mathematical specification to, and that failures of type inference are easily explained in terms of local features of the call site. In fact, many production programming languages such as C# and Java use a form of local type inference, due to the ease of extending this style of inference to support object-oriented features.

In this paper, we extend the local type inference algorithm to a language with both universal and existential quantifiers, which can both be instantiated impredicatively. This extended system is referred to as $\mathrm{F}^{\pm}_{\exists}$. The combination of features in $\mathrm{F}^{\pm}_{\exists}$ broke a number of the invariants which traditional type inference algorithms depend on and required us to invent new algorithms which combine both unification and (surprisingly) anti-unification.

We make the following ***contributions***:

Declarative Type System Specification (Sections 2 and 3) We present $\mathrm{F}^{\pm}_{\exists}$—a second-order polymorphic type system that supports first-class existential and universal quantifiers, both of which can be instantiated impredicatively. To evade the undecidability results surrounding subtyping for System F, we base the system on call-by-push-value (Levy, 2006): we use *polarization* of types and terms to formulate the restrictions that make typing decidable.

We give a *declarative* specification of local type inference in this system (i.e., the typing information is only available at a limited scope) based on a *subtyping* relation. Our specification makes it easy to see what is and what is not inferrable in the system.

To show that our restrictions do not unduly limit the expressiveness of the language, we prove that every $\mathrm{F}^{\pm}_{\exists}$ term can be embedded into System F and vice versa.

Type Inference Algorithm (Section 4) We give a type inference algorithm implementing the declarative specification. Because our system supports both existential and universal quantifiers, our implementation of the subtyping relation requires the use of *anti-unification*—a dual to unification that finds the most specific generalization rather than the most general unifier. To our knowledge, anti-unification has never been used in the context of polymorphic type inference.

The type inference algorithm itself is a purely local one, in the style of Pierce and Turner (2000). The locality can be seen from the fact that none of the unification variables introduced as part of type inference escape the scope of a single function application.

Correctness of the algorithm (Section 5) We prove that our algorithm is terminating and both sound and complete with respect to the declarative specification. This proof of soundness and completeness of the algorithm required us to introduce an intermediate system in-between the declarative and the algorithmic systems, and furthermore, the soundness and completeness proofs are mutually recursive with each other. The central theorems and the key proof ideas are formulated in Section 5, while the full definitions and formal proofs are deferred to the appendix (Appendices A and C).

Acknowledgments We would like to thank Meven Lennon-Bertrand for his extremely helpful comments and suggestions.

2 Overview

In the presence of impredicative polymorphism, System F has undecidable subtyping and type inference. It means that any inference algorithm is incomplete: it only infers the types for a *fragment* of System F.

To tackle this issue, we restrict System F by employing the ideas from the call-by-push-value system (Levy, 2006). We introduce a new language, $\mathrm{F}^{\pm}_{\exists}$, which stratifies the syntax of System F into positive and negative parts. It provides the structure that allows us to restrict the typing specification to make it decidable.

Every term and type of $\mathrm{F}^{\pm}_{\exists}$ has either *positive* or *negative* polarity. The type variables are annotated with their polarities (e.g. α^{+} or β^{-}), and the types can change polarity via the shift operators $\uparrow$ (positive to negative) and $\downarrow$ (negative to positive).

Positive expressions correspond to *values* in the call-by-push-value system, and negative expressions correspond to (potentially effectful) *computations*. The difference between the computations and values is intuitively described in the motto of call-by-push-value: "a value is, a computation does". In particular, a function type always takes a value and returns a computation. Similarly, the polymorphic $\forall$ quantifies a computation over positive types.

Whenever a computation is used in a value position (e.g. as an argument to a function), it must be *suspended* by a thunk constructor, which at the type level corresponds to the downshift operator $\downarrow N$. Symmetrically, to use a value as a computation, one can *return* it, which at the type level is reflected as an up-shift $\uparrow P$.

2.1 Examples

Generally, any term and type of System F can be embedded into $\mathrm{F}^{\pm}_{\exists}$ in multiple ways corresponding to different evaluation strategies of System F, which we will cover in more detail in Section 3.5. Figure 1 illustrates how standard System F types are polarized when call-by-value evaluation is presumed.

Observe the consistent pattern of the type polarization. Consider the type for `double`—a function multiplying an input integer by two. In System F, it would be $\texttt{Int} \to \texttt{Int}$. In call-by-value polarized form, it becomes $\downarrow(\texttt{Int} \to \uparrow\texttt{Int})$—a *suspended* function (indicated by

$$\begin{aligned}
\texttt{double} &: \downarrow(\texttt{Int} \to \uparrow\texttt{Int}) \\
\texttt{map} &: \downarrow(\forall\alpha^+.\, \forall\beta^+.\, \downarrow(\alpha^+ \to \uparrow\beta^+) \to [\alpha^+] \to \uparrow[\beta^+]) \\
\texttt{len} &: \downarrow(\forall\alpha^+.\, [\alpha^+] \to \uparrow\texttt{Int}) \\
\texttt{choose} &: \downarrow(\forall\alpha^+.\, \alpha^+ \to \alpha^+ \to \uparrow\alpha^+) \\
\texttt{id} &: \downarrow(\forall\alpha^+.\, \alpha^+ \to \uparrow\alpha^+) \\
\texttt{auto} &: \downarrow(\downarrow(\forall\alpha^+.\, \alpha^+ \to \uparrow\alpha^+) \to (\forall\alpha^+.\, \alpha^+ \to \uparrow\alpha^+))
\end{aligned}$$

Fig. 1: Polarization of System F types

the downshift) that accepts an integer `Int` and yields an integer *computation* (indicated by the upshift) $\uparrow\texttt{Int}$.

The type of `map` is more complex. Similar to `double`, the entire type is a suspended function, hence $\downarrow$. The function takes two implicit type arguments: α^+ and β^+, which indicated by the $\forall$s. The third argument of `map`—the applied function—has type $\alpha^+ \to \uparrow\beta^+$, which is suspended by $\downarrow$ since it is in a (positive) argument position. The last argument—$[\alpha^+]$—represents a list of α^+s and does not need to be suspended as it is already a value. Finally, the positive return type, $[\beta^+]$, is lifted to computations using $\uparrow$.

The types of `len`—the length function for lists, `choose`—a function that chooses between two input values, and `id`—the identity function, are similarly polarized. A slightly more complex function `auto` takes as an argument and returns something of the same identity type. However, since the argument position is positive, the argument type is suspended by $\downarrow$.

The types of `len` (the length function for lists), `choose` (a function that chooses between two input values), and `id` (the identity function) are polarized in a similar way. Another function, `auto`, takes an argument and returns a function of the same identity type. However, since the argument position is positive, the argument type is suspended using $\downarrow$.

Using the examples provided in Fig. 1, we now showcase the type inference capabilities and restrictions of $\mathrm{F}^{\pm}_{\exists}$.

Application of a Polymorphic Function A polymorphic function can be applied to a value of a concrete type. In this case, the explicit type application is not required: the inference algorithm automatically instantiates the quantified variables. For instance, the polymorphic `len` can be applied to a list of any positive type, and the resulting type will be inferred by the algorithm:

$$\cdot; \Gamma \vdash \texttt{let}\, x = \texttt{len}([1, 2, 3]);\ \texttt{return}\, x : \uparrow\texttt{Int}$$

Polymorphic Self-application. Unlike predicative systems, $\mathrm{F}^{\pm}_{\exists}$ permits self-application of polymorphic functions. For instance, `id` can be applied to itself resulting in a suspended polymorphic computation of type $\downarrow(\forall\alpha^+.\, \alpha^+ \to \uparrow\alpha^+)$, which, in turn, can be applied to itself:

$$\cdot; \Gamma \vdash \texttt{let}\, x = \texttt{id}(\texttt{id});\ \texttt{let}\, y = x(x);\ \texttt{return}\, y : \uparrow\downarrow(\forall\alpha^+.\, \alpha^+ \to \uparrow\alpha^+)$$

Negative declarative types

$$N, M, K \quad ::= \quad \alpha^- \mid \uparrow P \mid P \to N \mid \forall \overrightarrow{\alpha^+}.\, N$$

Positive declarative types

$$P, Q, R \quad ::= \quad \alpha^+ \mid \downarrow N \mid \exists \overrightarrow{\alpha^-}.\, P$$

Fig. 2: Declarative Types of $\mathrm{F}^{\pm}_{\exists}$

Non-example: Polymorphic Arguments The focus of our system is the *local* type inference (Pierce and Turner, 2000), which has certain limitations. In particular, the instantiation does not happen when the polymorphic term is in the argument position. For example, one could expect the following to be inferrable: $\cdot; \Gamma \vdash \mathtt{let}\, x = \mathtt{map}(\mathtt{id}, [2,3,9]); \mathtt{return}\, x : \uparrow[\mathtt{Int}]$. However, to infer this, `id` must be instantiated to $\mathtt{Int} \to \uparrow\mathtt{Int}$ which requires the information to be propagated from the neighboring branch of the syntax tree ($[2,3,9]$), i.e., bypass the locality.

Inferring the Common Supertype The function `choose` takes two arguments of the same polymorphic type. To apply it to two values of different (concrete) types, the inference algorithm must find the minimal type they can be coerced to—the least common supertype.

Let us assume that in Γ, we have two identity-like functions $\mathtt{id}_M : \downarrow(\downarrow M \to M)$ and $\mathtt{id}_N : \downarrow(\downarrow N \to N)$. Then their least common supertype—the existential $\exists \gamma^-.\, \downarrow(\downarrow \gamma^- \to \gamma^-)$ is inferrable:

$$\cdot; \Gamma \vdash \mathtt{let}\, x = \mathtt{choose}(\mathtt{id}_N, \mathtt{id}_M); \mathtt{return}\, x : \uparrow \exists \gamma^-.\, \downarrow(\downarrow \gamma^- \to \gamma^-)$$

Inferring a general type As mentioned before, the local inference does not instantiate the polymorphic types in the argument position. Because of that, the application $\mathtt{let}\, x = \mathtt{choose}(\mathtt{id}, \mathtt{auto}); \mathtt{return}\, x$ cannot infer $\uparrow\downarrow(\downarrow(\forall \alpha^+.\, \alpha^+ \to \uparrow\alpha^+) \to (\forall \alpha^+.\, \alpha^+ \to \uparrow\alpha^+))$. However, the inference succeeds and infers a less informative type:

$$\cdot; \Gamma \vdash \mathtt{let}\, x = \mathtt{choose}(\mathtt{id}, \mathtt{auto}); \mathtt{return}\, x : \uparrow \exists \gamma^-.\, \downarrow \gamma^-$$

2.2 The Language of Types

The types of $\mathrm{F}^{\pm}_{\exists}$ are given in Fig. 2. They are stratified into two syntactic categories (polarities): positive and negative, similar to the values and computations in the call-by-push-value system.

- $\pm$ α^+ and α^- denote negative and positive type variables, respectively. The variables are either introduced by the corresponding quantifiers ($\forall$ and $\exists$, respectively) or declared in the type context;
- $-$ a function $P \to N$ takes a *positive* input P and returns a *negative* output N; the function type itself is *negative*, so the 'arrow' operator is right-associative: $P \to Q \to N$ is equivalent to $P \to (Q \to N)$;
- $-$ similarly to functions, a polymorphic abstraction $\forall \overrightarrow{\alpha^+}.\, N$ quantifies a negative type N over a list of positive type variables $\overrightarrow{\alpha^+}$;
- $+$ dually, $\exists \overrightarrow{\alpha^-}.\, P$ binds negative variables in a positive type P;

Computation Terms

$$c,\ d \quad ::= \quad (c : N) \mid \lambda x : P.\ c \mid \Lambda\alpha^{+}.\ c \mid \texttt{return}\ v \mid \texttt{let}\ x = v;\ c \mid \texttt{let}\ x : P = c;\ c' \mid \texttt{let}\ x : P = v(\overrightarrow{v});\ c \mid \texttt{let}\ x = v(\overrightarrow{v});\ c \mid \texttt{let}^{\exists}(\overrightarrow{\alpha^{-}}, x) = v;\ c$$

Value Terms

$$v,\ w \quad ::= \quad x \mid \{c\} \mid (v : P)$$

Fig. 3: Declarative Terms of $\mathrm{F}^{\pm}_{\exists}$

± an upshift $\uparrow P$ and a downshift $\downarrow N$ change the polarity of a type from positive to negative and vice versa. At the level of terms, the constructors for shift types are `return` v and $\{c\}$ (thunked computation), respectively.

Syntactic Conventions We always consider terms and types up to alpha-equivalence, and substitution is capture-avoiding. In addition, we assume the quantification is associative, for example, $\forall\overrightarrow{\alpha^{+}}.\ \forall\overrightarrow{\beta^{+}}.\ N$ and $\forall\overrightarrow{\alpha^{+}},\ \overrightarrow{\beta^{+}}.\ N$ represent the same type.

Type Context and Type Well-formedness In the construction of $\mathrm{F}^{\pm}_{\exists}$, a type context (denoted as Θ) is represented as a *set* of positive and negative type variables and it is used to assert the well-formedness of types. The well-formedness (or well-scopeness) of a type is denoted as $\Theta \vdash P$ and $\Theta \vdash N$ and it asserts that all type variables are either bound by a quantifier ($\forall$ and $\exists$) or declared in the context Θ. The well-formedness checking is an *algorithmic* procedure. We represent it as a system of inference rules, that correspond to a recursive algorithm taking the context and the type as input. For brevity, we omit these rules, as they are standard binder scoping rules.

2.3 The Language of Terms

In Fig. 3, we define the language of terms of $\mathrm{F}^{\pm}_{\exists}$. The language is a combination of System F and call-by-push-value constructs: the terms are stratified into values and computations; there are return and thunk constructors, as well as several forms of let bindings to sequence computations, bind function applications, and eliminate existentials, all with or without type annotations.

+ x denotes a term variable. Following the call-by-push-value stratification, we only have *positive* (value) term variables;
+ $\{c\}$ is a value corresponding to a thunked or suspended computation;
± $(c : N)$ and $(v : P)$ allow one to annotate positive and negative terms;
– `return` v is a pure computation, returning a value;
– $\lambda x : P.\ c$ and $\Lambda\alpha^{+}.\ c$ are standard lambda abstractions. Notice that we require the type annotation for the argument of λ;
– `let` $x = v;\ c$ is a standard let form, binding a value v to a variable x in a computation c;
– computational let form `let` $x : P = c;\ c'$ operates similarly to a monadic bind. It binds the result of a computation $c : \uparrow P$ to a variable $x : P$, and continues with a computation c';

- applicative let forms $\texttt{let}\, x : P = v(\vec{v});\ c$ and $\texttt{let}\, x = v(\vec{v});\ c$ can be viewed as a special case of the computational let, when the first computation is a function application $v(\vec{v})$. They take a function v, apply it to a list of arguments, bind the (pure) result to a variable x, and continues with a computation c. If the resulting type of the application is unique, one can omit the type annotation, as in the second form: it will be inferred by the algorithm;
- $\texttt{let}^{\exists}(\overrightarrow{\alpha^-}, x) = v;\ c$ is the standard eliminator of an existential type (unpack): expecting v to be of an existential type, it binds the packed negative types to a list of variables $\overrightarrow{\alpha^-}$, binds the body of the existential to x, and continues with a computation c.

Missing constructors Notice that the language does not have first-class applications: their role is played by the applicative let forms, binding the result of a *fully applied* function to a variable. Also notice that the language does not have a type application (i.e. the eliminator of $\forall$) and dually, it does not have `pack` (i.e. the constructor of $\exists$). This is because the instantiation of polymorphic and existential types is inferred by the algorithm. In Section 6, we discuss the way to modify the system to introduce *explicit* type applications.

2.4 The Key Ideas of the Algorithm

The inference algorithm for $\mathrm{F}^{\pm}_{\exists}$ is *local*: it has a limited scope of the inference information and does not propagate it between far-apart branches of the syntax tree. Nevertheless, many difficulties appear in the inference of the combination of polymorphic and existential types. We now discuss the most challenging of these difficulties and how they are addressed in the algorithm.

Subtyping The inference algorithm's complexity mainly lies in the subtyping relation, which is responsible for polymorphic instantiation. It is this subtyping relation that enables us to apply polymorphic terms in situations where their specific instantiations are anticipated.

To check whether one type is a subtype of another, the algorithm introduces *algorithmic* type variables ($\widehat{\alpha}^{\pm}, \widehat{\beta}^{\pm}, \ldots$)—the 'placeholders' representing the polymorphic variables whose instantiation is postponed. This way, the problem of $\forall\alpha^+.\uparrow\alpha^+ \leqslant \uparrow\texttt{Int}$ becomes the problem of finding $\widehat{\alpha}^+$ such that $\uparrow\widehat{\alpha}^+ \leqslant \uparrow\texttt{Int}$.

Equivalence The system introduces another complexity: the equivalence induced by subtyping ($\Theta \vdash N \simeq^{\leqslant} M \overset{\text{def}}{\iff} \Theta \vdash N \leqslant M$ and $\Theta \vdash M \leqslant N$) is not straightforward. This relation extends beyond just alpha-equivalent types. As the neighboring polymorphic quantifiers can be instantiated in arbitrary order, the equivalence relation must permit the *permutations* of the quantifiers. For example, we have $\cdot \vdash \forall\alpha^+.\forall\beta^+.N \simeq^{\leqslant} \forall\beta^+.\forall\alpha^+.N$. Similarly, the equivalence permits the *removal/addition* of the unused quantifiers: $\cdot \vdash \forall\alpha^+.\uparrow\texttt{Int} \simeq^{\leqslant} \uparrow\texttt{Int}$.

To handle the complexity of non-trivial equivalence, we employ a type normalization algorithm. This approach simplifies mutual subtyping to a more manageable alpha-equivalence by eliminating unused quantifiers and reordering the rest into a canonical permutation, effectively replacing subtyping-induced equivalence on terms with alpha-equivalence on their normal forms.

Least Upper Bound Eventually, the subtyping algorithm reduces the initial subtyping problem to a set of constraints that need to be resolved. At this moment, another problem arises: as one variable can occur in multiple constraints, the resolution becomes non-trivial. In particular, to resolve the conjunction of $\widehat{\alpha}^{+} \geqslant P$ and $\widehat{\alpha}^{+} \geqslant Q$, we would need to find the least upper bound of P and Q.

In Damas Milner type inference, the least upper bound boils down to the trivial syntactic equality. Because of that, the set of constraints is typically resolved using *unification*. However, this problem is trickier in the presence of existential types. For example, in $\mathrm{F}^{\pm}_{\exists}$, the least upper bound of $\downarrow\uparrow\texttt{Int}$ and $\downarrow\uparrow\texttt{Bool}$ is $\exists\alpha^{-}.\downarrow\alpha^{-}$, because the quantified α^{-} can be instantiated to either $\uparrow\texttt{Int}$ or $\uparrow\texttt{Bool}$.

To find this least upper bound, we use *anti-unification*—the dual of unification. While unification finds *the most general instance* of two given patterns, the anti-unification finds *the most detailed* (under the restrictions of the system) *pattern* that matches two given types. In the example above, the anti-unifier of $\downarrow\uparrow\texttt{Int}$ and $\downarrow\uparrow\texttt{Bool}$ is $\downarrow\widehat{\alpha}^{-}$, and in $\mathrm{F}^{\pm}_{\exists}$, this anti-unifier is the most specific (in particular because $\downarrow\uparrow\widehat{\alpha}^{+}$ is not considered as a candidate since only *negative* variables are existentially quantified).

Impredicativity Another difficulty that hinders the constraint resolution arises from the impredicativity of the system—the ability to quantify over the types that are themselves polymorphic. In the impredicative polymorphic system (even with only $\forall$-quantifiers), the naturally formulated subtyping is undecidable, which follows from the undecidability of second-order unification. All the more the undecidability is evident in the presence of existential types and subtyping constraints. For instance, in $\mathrm{F}^{\pm}_{\exists}$ the constraint $\widehat{\alpha}^{+} :\geqslant \downarrow\uparrow\widehat{\alpha}^{+}$ would have a surprising impredicative solution $\widehat{\alpha}^{+} = \exists\alpha^{-}.\downarrow\alpha^{-}$.

To maintain decidability in the presence of impredicativity, we carefully restrict the system using the polarity stratification. One of the main constraints that we put on subtyping is the invariance of the shift operators: the subtyping $\uparrow P \leqslant \uparrow Q$ is only allowed if $Q \geqslant P$ *and* $P \geqslant Q$. These constraints do not trivialize the system: we still have non-trivial upper bounds, and thus, need to use anti-unification to find them. However, it prevents such examples as $\widehat{\alpha}^{+} :\geqslant \downarrow\uparrow\widehat{\alpha}^{+}$, and as we prove, makes the constraint resolution decidable.

3 Declarative System

In this section, we present the declarative system of $\mathrm{F}^{\pm}_{\exists}$, which serves as a specification of the type inference algorithm. The declarative system is represented as a set of inference rules and consists of two main subsystems: subtyping and type inference. First, we present the declarative subtyping rules specifying when one type is a subtype of another. Next, we discuss the equivalence relation induced by mutual subtyping. Finally, we present the type inference rules, that refer to the subtyping and equivalence relation. We conclude this section by discussing the relation between the proposed type system and the standard System F.

$\boxed{\Theta \vdash N \leqslant M}$ Negative subtyping

$$\frac{}{\Theta \vdash \alpha^- \leqslant \alpha^-}\ (\textsc{Var}^{\leqslant}_{-})$$

$$\frac{\Theta \vdash P \simeq^{\leqslant} Q}{\Theta \vdash \uparrow P \leqslant \uparrow Q}\ (\uparrow^{\leqslant})$$

$$\frac{\Theta \vdash P \geqslant Q \quad \Theta \vdash N \leqslant M}{\Theta \vdash P \to N \leqslant Q \to M}\ (\to^{\leqslant})$$

$$\frac{\Theta, \overrightarrow{\beta^+} \vdash \sigma : \overrightarrow{\alpha^+} \quad \Theta, \overrightarrow{\beta^+} \vdash [\sigma]N \leqslant M}{\Theta \vdash \forall \overrightarrow{\alpha^+}.\ N \leqslant \forall \overrightarrow{\beta^+}.\ M}\ (\forall^{\leqslant})$$

$\boxed{\Theta \vdash P \geqslant Q}$ Positive supertyping

$$\frac{}{\Theta \vdash \alpha^+ \geqslant \alpha^+}\ (\textsc{Var}^{\geqslant}_{+})$$

$$\frac{\Theta \vdash N \simeq^{\leqslant} M}{\Theta \vdash \downarrow N \geqslant \downarrow M}\ (\downarrow^{\geqslant})$$

$$\frac{\Theta, \overrightarrow{\beta^-} \vdash \sigma : \overrightarrow{\alpha^-} \quad \Theta, \overrightarrow{\beta^-} \vdash [\sigma]P \geqslant Q}{\Theta \vdash \exists \overrightarrow{\alpha^-}.\ P \geqslant \exists \overrightarrow{\beta^-}.\ Q}\ (\exists^{\geqslant})$$

$\boxed{\Theta \vdash N \simeq^{\leqslant} M}$ Negative equivalence

$$\frac{\Theta \vdash N \leqslant M \quad \Theta \vdash M \leqslant N}{\Theta \vdash N \simeq^{\leqslant} M}\ (\simeq^{\leqslant}_{-})$$

$\boxed{\Theta \vdash P \simeq^{\leqslant} Q}$ Positive equivalence

$$\frac{\Theta \vdash P \geqslant Q \quad \Theta \vdash Q \geqslant P}{\Theta \vdash P \simeq^{\leqslant} Q}\ (\simeq^{\leqslant}_{+})$$

Fig. 4: Declarative Subtyping

3.1 Subtyping

The inference rules representing declarative subtyping are shown in Fig. 4. Let us discuss them in more detail.

Quantifiers Symmetric rules $(\forall^{\leqslant})$ and $(\exists^{\geqslant})$ specify subtyping between top-level quantified types. Usually, the polymorphic subtyping is represented by two rules introducing quantifiers to the left and to the right-hand side of subtyping. For conciseness of representation, we compose these rules into one. First, our rule extends context Θ with the quantified variables from the right-hand side ($\overrightarrow{\beta^+}$ or $\overrightarrow{\beta^-}$), as these variables must remain abstract. Second, it verifies that the left-hand side quantifiers ($\overrightarrow{\alpha^+}$ or $\overrightarrow{\alpha^-}$) can be instantiated with an arbitrary substitution to continue subtyping recursively, which introduces non-determinism.

The instantiation of quantifiers is modeled by a substitution σ. The notation $\Theta_2 \vdash \sigma : \Theta_1$ specifies its domain and range. For instance, $\Theta, \overrightarrow{\beta^+} \vdash \sigma : \overrightarrow{\alpha^+}$ means that σ maps the variables from $\overrightarrow{\alpha^+}$ to (positive) types well-formed in $\Theta, \overrightarrow{\beta^+}$. This way, application $[\sigma]N$ instantiates (replaces) every α_i^- in N with $\sigma(\alpha_i^-)$.

Invariant Shifts As mentioned above, an important restriction that we put on the subtyping system is that subtyping of shifted types requires their equivalence, as shown in $(\downarrow^{\geqslant})$ and $(\uparrow^{\leqslant})$. The reason for this is that if both of these rules were relaxed to the covariant form, the subtyping relation would become equivalent to the standard subtyping of System F, which is undecidable (Tiuryn and Urzyczyn, 1996). Relaxations of this condition are discussed in Section 6.2.

Functions Standardly, subtyping of function types is covariant in the return type and contravariant in the argument type.

Variables Subtyping of variables is defined reflexively, which is enough to ensure the reflexivity of subtyping in general. The algorithm—specifically the least upper bound

procedure—will use the fact that the subtypes of a variable coincide with its supertypes (Property 2), which however does not hold for arbitrary types.

3.2 Properties of Declarative Subtyping

A property that is important for the subtyping algorithm, in particular for the type *upgrade* procedure (Section 4.6), is the preservation of free variables by subtyping. Informally, it says that the free variables of a positive type cannot disappear in its subtypes, and the free variables of a negative type cannot disappear in its supertypes.

Property 1 (Subtyping preserves free variables)**.** *Let us assume that all the mentioned types are well-formed in* Θ*. Then* $\Theta \vdash N_1 \leqslant N_2$ *implies* $\mathtt{fv}\,(N_1) \subseteq \mathtt{fv}\,(N_2)$*, and* $\Theta \vdash P_1 \geqslant P_2$ *implies* $\mathtt{fv}\,(P_1) \subseteq \mathtt{fv}\,(P_2)$*.*

Property 2 (Variable subtyping is trivial)**.** *A subtype or a supertype of a variable is equivalent to the variable itself:*

$$- \quad \Theta \vdash \alpha^- \leqslant N \iff \Theta \vdash N \leqslant \alpha^- \iff N = \forall \overrightarrow{\beta^+}.\, \alpha^-$$
$$+ \quad \Theta \vdash \alpha^+ \geqslant P \iff \Theta \vdash P \geqslant \alpha^+ \iff P = \exists \overrightarrow{\beta^-}.\, \alpha^+$$

Another property that we extensively use is that subtyping is reflexive and transitive, and agrees with substitution.

Property 3 (Subtyping forms a preorder)**.** *For a fixed context* Θ*, the negative subtyping relation* $\Theta \vdash N_1 \leqslant N_2$ *and the positive subtyping relation* $\Theta \vdash P_1 \geqslant P_2$ *are reflexive and transitive on types well-formed in* Θ*.*

Property 4 (Subtyping agrees with substitution)**.** *Suppose that* σ *is a substitution such that* $\Theta_2 \vdash \sigma : \Theta_1$*. Then*

- $\Theta_1 \vdash N \leqslant M$ *implies* $\Theta_2 \vdash [\sigma]N \leqslant [\sigma]M$*, and*
- $\Theta_1 \vdash P \geqslant Q$ *implies* $\Theta_2 \vdash [\sigma]P \geqslant [\sigma]Q$*.*

Moreover, any two *positive* types have a least upper bound, which makes positive subtyping a semilattice. The positive least upper bound can be found algorithmically, which we will discuss in the next section.

Property 5 (Positive least upper bound exists)**.** *Suppose that* P_1 *and* P_2 *are positive types well-formed in* Θ*. Then there exists a least common supertype—a type* P *such that*

- $\Theta \vdash P \geqslant P_1$ *and* $\Theta \vdash P \geqslant P_2$*, and*
- *for any* Q *such that* $\Theta \vdash Q \geqslant P_1$ *and* $\Theta \vdash Q \geqslant P_2$*,* $\Theta \vdash Q \geqslant P$*.*

Negative greatest lower bound might not exist The symmetric construction—the greatest lower bound of two negative types—does not always exist, as the following counterexample shows. Consider the following types:

- N and Q are arbitrary closed types,
- P, P_1, and P_2 are non-equivalent closed types such that $\cdot \vdash P_1 \geqslant P$ and $\cdot \vdash P_2 \geqslant P$, and none of the types is equivalent to Q.

What is the greatest common subtype of $Q \to \downarrow\uparrow Q \to \downarrow\uparrow Q \to N$ and $P \to \downarrow\uparrow P_1 \to \downarrow\uparrow P_2 \to N$? The type $\forall \alpha^+, \beta^+, \gamma^+.\ \alpha^+ \to \downarrow\uparrow \beta^+ \to \downarrow\uparrow \gamma^+ \to N$ is a common subtype, however, it is not the greatest one, as it is too general.

One can find two greater candidates: $M_1 = \forall \alpha^+, \beta^+.\ \alpha^+ \to \downarrow\uparrow \alpha^+ \to \downarrow\uparrow \beta^+ \to N$ and $M_2 = \forall \alpha^+, \beta^+.\ \beta^+ \to \downarrow\uparrow \alpha^+ \to \downarrow\uparrow \beta^+ \to N$. Instantiating α^+ and β^+ with Q ensures that both of these types are subtypes of $Q \to \downarrow\uparrow Q \to \downarrow\uparrow Q \to N$; instantiating α^+ with P_1 and β^+ with P_2 demonstrates the subtyping with $P \to \downarrow\uparrow P_1 \to \downarrow\uparrow P_2 \to N$, as P is a subtype of both P_1 and P_2.

By analyzing the inference rules, we can prove that both M_1 and M_2 are *maximal* common subtypes, i.e., there is no common subtype that is greater than them. However, M_1 and M_2 are not equivalent, which means that none of them is the greatest.

3.3 Equivalence and Normalization

The subtyping relation forms a preorder on types, and thus, it induces an equivalence relation also known as bicoercibility (Tiuryn, 1995). The declarative specification of subtyping must be defined up to this equivalence. Moreover, the algorithms we use must withstand changes in input types within the equivalence class. To deal with non-trivial equivalence, we use normalization—a function that uniformly selects a representative of the equivalence class.

Using normalization gives us two benefits: (i) we do not need to modify significantly standard operations such as unification to withstand non-trivial equivalence, and (ii) if the subtyping (and thus, the equivalence) changes, we only need to modify the normalization function, while the rest of the algorithm remains the same.

In our system, equivalence is richer than equality. Specifically, while staying within one equivalence class, one can change the type:

(i) introduce and remove redundant quantifiers. For example, $\forall \alpha^+, \beta^+.\ \uparrow \alpha^+$ is equivalent but not equal to $\forall \alpha^+.\ \uparrow \alpha^+$;

(ii) reorder adjacent quantifiers. For example, $\forall \alpha^+, \beta^+.\ \alpha^+ \to \beta^+ \to \gamma^-$ is equivalent but not equal to $\forall \alpha^+, \beta^+.\ \beta^+ \to \alpha^+ \to \gamma^-$;

(iii) make the transformations (i) and (ii) at any position in the type.

It turns out that the transformations (i-iii) are complete, in the sense that they generate the whole equivalence class. This way, to normalize the type, one must

(i) remove the redundant quantifiers,

(ii) reorder the quantifiers to the canonical order,

(iii) do the procedures (i) and (ii) recursively on the subterms.

The normalization algorithm is shown in Fig. 5. The steps (i-ii) are implemented by the *ordering* function '`ord` *vars* `in` N' and '`ord` *vars* `in` P'. For a set of variables *vars* and a type, it returns a list of variables from *vars* that occur in the type in the *order of their first*

$\boxed{\mathtt{nf}\,(N) = M}$

$$\frac{}{\mathtt{nf}\,(\alpha^-) = \alpha^-}\ (\textsc{Var}_-^{\textsc{nf}})$$

$$\frac{\mathtt{nf}\,(P) = Q}{\mathtt{nf}\,(\uparrow P) = \uparrow Q}\ (\uparrow^{\textsc{nf}})$$

$$\frac{\mathtt{nf}\,(P) = Q \quad \mathtt{nf}\,(N) = M}{\mathtt{nf}\,(P \to N) = Q \to M}\ (\to^{\textsc{nf}})$$

$$\frac{\mathtt{nf}\,(N) = N' \quad \mathtt{ord}\,\overrightarrow{\alpha^+}\,\mathtt{in}\,N' = \overrightarrow{\alpha^+}'}{\mathtt{nf}\,(\forall\overrightarrow{\alpha^+}.\,N) = \forall\overrightarrow{\alpha^+}'.\,N'}\ (\forall^{\textsc{nf}})$$

$\mathtt{ord}\,\mathit{vars}\,\mathtt{in}\,N$ returns a list of variables from $\mathit{vars} \cap \mathtt{fv}\,(N)$ in the order of their first occurrence in N

$\boxed{\mathtt{nf}\,(P) = Q}$

$$\frac{}{\mathtt{nf}\,(\alpha^+) = \alpha^+}\ (\textsc{Var}_+^{\textsc{nf}})$$

$$\frac{\mathtt{nf}\,(N) = M}{\mathtt{nf}\,(\downarrow N) = \downarrow M}\ (\downarrow^{\textsc{nf}})$$

$$\frac{\mathtt{nf}\,(P) = P' \quad \mathtt{ord}\,\overrightarrow{\alpha^-}\,\mathtt{in}\,P' = \overrightarrow{\alpha^-}'}{\mathtt{nf}\,(\exists\overrightarrow{\alpha^-}.\,P) = \exists\overrightarrow{\alpha^-}'.\,P'}\ (\exists^{\textsc{nf}})$$

$\mathtt{ord}\,\mathit{vars}\,\mathtt{in}\,P$ returns a list of variables from $\mathit{vars} \cap \mathtt{fv}\,(P)$ in the order of their first occurrence in P

Fig. 5: Type Normalization Procedure

occurrence. For brevity, we omit the formal definition of ordering referring the reader to the appendix (Algorithm 4).

For the normalization procedure, we prove soundness and completeness w.r.t. the equivalence relation.

Property 6 (Correctness of normalization)**.** *Assuming all types are well-formed in* Θ*,*

- $\Theta \vdash N \simeq^{\leqslant} M$ *is equivalent to* $\mathtt{nf}\,(N) = \mathtt{nf}\,(M)$*, and*
- $\Theta \vdash P \simeq^{\leqslant} Q$ *is equivalent to* $\mathtt{nf}\,(P) = \mathtt{nf}\,(Q)$.

3.4 Typing

The declarative specification of the type inference is given in Fig. 6. It consists of three main judgments: the positive typing judgment, the negative typing judgment, and the application typing judgment. The positive typing judgment $\Theta; \Gamma \vdash v\colon P$ is read as "under the type context Θ and variable context Γ, the term v is allowed to infer type P", where Γ—the variable context—is defined standardly as a set of pairs of the form $x : P$. The negative typing judgment is read similarly. The *Application typing* judgment infers the type of the application of a function to a list of arguments. It has form $\Theta\,; \Gamma \vdash N \bullet \overrightarrow{v} \Rrightarrow M$, which reads "under the type context Θ and variable context Γ, the application of a function of type N to the list of arguments $\overrightarrow{v}$ is allowed to infer type M".

Let us discuss the rules of the declarative system in more detail.

Variables Rule $(\textsc{Var}^{\textsc{inf}})$ allows us to infer the type of a variable from the context. In literature, one can find another version of this rule, that enables inferring a type *equivalent* to the type from the context. In our case, the inference of equivalent types is admissible by $(\simeq_+^{\textsc{inf}})$.

Annotations The annotation rules $(\textsc{ann}_-^{\textsc{inf}})$ and $(\textsc{ann}_+^{\textsc{inf}})$ use subtyping. The annotation is only valid if the inferred type is a subtype of the annotation type.

$\boxed{\Theta; \Gamma \vdash c : N}$ Negative typing

$$\frac{\Theta \vdash P \quad \Theta; \Gamma, x : P \vdash c : N}{\Theta; \Gamma \vdash \lambda x : P.\, c : P \to N}\ (\lambda^{\text{INF}})$$

$$\frac{\Theta, \alpha^+; \Gamma \vdash c : N}{\Theta; \Gamma \vdash \Lambda \alpha^+.\, c : \forall \alpha^+.\, N}\ (\Lambda^{\text{INF}})$$

$$\frac{\Theta; \Gamma \vdash v : P}{\Theta; \Gamma \vdash \mathtt{return}\ v : \uparrow P}\ (\text{RET}^{\text{INF}})$$

$$\frac{\Theta; \Gamma \vdash v : P \quad \Theta; \Gamma, x : P \vdash c : N}{\Theta; \Gamma \vdash \mathtt{let}\ x = v;\ c : N}\ (\text{LET}^{\text{INF}})$$

$$\frac{\Theta \vdash P \quad \Theta; \Gamma \vdash c : M \quad \Theta \vdash M \leqslant \uparrow P \quad \Theta; \Gamma, x : P \vdash c' : N}{\Theta; \Gamma \vdash \mathtt{let}\ x : P = c;\ c' : N}\ (\text{LET}_{\text{C}}^{\text{INF}})$$

$$\frac{\Theta; \Gamma \vdash c : N \quad \Theta \vdash N \simeq^{\leqslant} N'}{\Theta; \Gamma \vdash c : N'}\ (\simeq_{-}^{\text{INF}})$$

$$\frac{\Theta; \Gamma \vdash v : \downarrow M \quad \Theta ; \Gamma \vdash M \bullet \vec{v} \Longrightarrow \uparrow Q \text{ principal} \quad \Theta; \Gamma, x : Q \vdash c : N}{\Theta; \Gamma \vdash \mathtt{let}\ x = v(\vec{v});\ c : N}\ (\text{LET}_{@}^{\text{INF}})$$

$$\frac{\Theta \vdash P \quad \Theta; \Gamma \vdash v : \downarrow M \quad \Theta ; \Gamma \vdash M \bullet \vec{v} \Longrightarrow M' \quad \Theta \vdash M' \leqslant \uparrow P \quad \Theta; \Gamma, x : P \vdash c : N}{\Theta; \Gamma \vdash \mathtt{let}\ x : P = v(\vec{v});\ c : N}\ (\text{LET}_{:@}^{\text{INF}})$$

$$\frac{\Theta; \Gamma \vdash v : \exists \overrightarrow{\alpha^-}.\, P \quad \mathtt{nf}\ (\exists \overrightarrow{\alpha^-}.\, P) = \exists \overrightarrow{\alpha^-}.\, P \quad \Theta, \overrightarrow{\alpha^-}; \Gamma, x : P \vdash c : N \quad \Theta \vdash N}{\Theta; \Gamma \vdash \mathtt{let}^{\exists}(\overrightarrow{\alpha^-}, x) = v;\ c : N}\ (\text{LET}_{\exists}^{\text{INF}})$$

$$\frac{\Theta \vdash M \quad \Theta; \Gamma \vdash c : N \quad \Theta \vdash N \leqslant M}{\Theta; \Gamma \vdash (c : M) : M}\ (\text{ANN}_{-}^{\text{INF}})$$

$\boxed{\Theta; \Gamma \vdash v : P}$ Positive typing

$$\frac{x : P \in \Gamma}{\Theta; \Gamma \vdash x : P}\ (\text{VAR}^{\text{INF}})$$

$$\frac{\Theta; \Gamma \vdash c : N}{\Theta; \Gamma \vdash \{c\} : \downarrow N}\ (\{\}^{\text{INF}})$$

$$\frac{\Theta \vdash Q \quad \Theta; \Gamma \vdash v : P \quad \Theta \vdash Q \geqslant P}{\Theta; \Gamma \vdash (v : Q) : Q}\ (\text{ANN}_{+}^{\text{INF}})$$

$$\frac{\Theta; \Gamma \vdash v : P \quad \Theta \vdash P \simeq^{\leqslant} P'}{\Theta; \Gamma \vdash v : P'}\ (\simeq_{+}^{\text{INF}})$$

$\boxed{\Theta ; \Gamma \vdash N \bullet \vec{v} \Longrightarrow M}$ Application typing

$$\frac{\Theta \vdash N \simeq^{\leqslant} N'}{\Theta ; \Gamma \vdash N \bullet \cdot \Longrightarrow N'}\ (\emptyset_{\bullet\Rightarrow}^{\text{INF}})$$

$$\frac{\Theta; \Gamma \vdash v : P \quad \Theta \vdash Q \geqslant P \quad \Theta ; \Gamma \vdash N \bullet \vec{v} \Longrightarrow M}{\Theta ; \Gamma \vdash Q \to N \bullet v, \vec{v} \Longrightarrow M}\ (\to_{\bullet\Rightarrow}^{\text{INF}})$$

$$\frac{\vec{v} \neq \quad \overrightarrow{\alpha^+} \neq \cdot \quad \Theta \vdash \sigma : \overrightarrow{\alpha^+} \quad \Theta ; \Gamma \vdash [\sigma] N \bullet \vec{v} \Longrightarrow M}{\Theta ; \Gamma \vdash \forall \overrightarrow{\alpha^+}.\, N \bullet \vec{v} \Longrightarrow M}\ (\forall_{\bullet\Rightarrow}^{\text{INF}})$$

Fig. 6: Declarative Inference

Abstractions The typing of lambda abstraction is standard. Rule (λ^{INF}) first checks that the given type P annotating the argument is well-formed, and then infers the type N of the body in the extended context. As a result, it returns an arrow type $P \to N$. Rule (Λ^{INF}) infers polymorphic $\forall$-type. It extends the type context with the quantifying variable α^+ and infers the type N of the body. As a result, it returns a polymorphic type $\forall \alpha^+.\, N$.

Return and thunk Rules $(\text{RET}^{\text{INF}})$ and $(\{\}^{\text{INF}})$ simply add the corresponding shift constructors to the type of the body.

Unpack Rule $(\text{LET}^{\text{INF}}_{\exists})$ types elimination of $\exists$. First, it infers the normalized type of the existential package. The normalization is required to fix the order of the quantifying variables to bind them. Alternative approaches that do not require normalization will be discussed in Section 6.1. After the bind, the rule infers the type of the body and ensures that the bound variables do not escape the scope.

Let binders Rule $(\text{LET}^{\text{INF}})$ is a standard rule for typing let binders: we infer the type of the bound value and continue the typing of the computation in the extended context.

Rule $(\text{LET}^{\text{INF}}_{\text{C}})$ associates a variable with a computation. The inferred type for this computation (M) must be castable to a shifted positive type $\uparrow P$, with P then assigned to the bound variable x to type the let binder's body. Like all annotated constructors, we also verify the annotation type P's well-formedness.

Applicative let binders Rules $(\text{LET}^{\text{INF}}_{@})$ and $(\text{LET}^{\text{INF}}_{:@})$ infer the type of the applicative let binders. Both of them infer the type of the head v and invoke the application typing to infer the type of the application before recursing on the body of the let binder.

The former rule infers the type of an *unannotated* let binder, and thus it requires the resulting type of application to be *the principal type*, so that the type we assign to the bound variable x is determined. In this context, principality means minimality. In other words, $\Theta\,;\Gamma \vdash M \bullet \vec{v} \Longrightarrow \uparrow Q$ principal means that any other type Q' inferrable for the application (i.e., $\Theta\,;\Gamma \vdash M \bullet \vec{v} \Longrightarrow \uparrow Q'$) is greater than the principal type Q, i.e., $\Theta \vdash Q' \geqslant Q$.

The latter rule is for the *annotated* binder, and thus, the type of the bound x is given, however, the rule must check that this type is a a supertype of the inferred type of the application. This check is done by invoking the subtyping judgment $\Theta \vdash M' \leqslant \uparrow P$.

Typing up to equivalence As discussed in Section 3.3, the subtyping, as a preorder, induces a non-trivial equivalence relation on types. The system must not distinguish between equivalent types, and thus, type inference must be defined up to equivalence. For this purpose, we use rules $(\simeq^{\text{INF}}_{+})$ and $(\simeq^{\text{INF}}_{-})$. They allow one to replace the inferred type with an equivalent one.

Application to an empty list of arguments The base case of the application type inference is represented by rule $(\emptyset^{\text{INF}}_{\bullet\Longrightarrow})$. If the head of the type N is applied to no arguments, the type of the result is allowed to be N or any equivalent type. We need to relax this rule up to equivalence to ensure the corresponding property globally: the inferred application type can be replaced with an equivalent one. Alternatively, we could have added a separate rule similar to $(\simeq^{\text{INF}}_{+})$, however, the local relaxation is sufficient to prove the global property.

Application of a polymorphic type $\forall$ The complexity of the system lives in the rules whose output type is not immediately defined by their input and the output of their premises (also known as not mode-correct (Dunfield and Krishnaswami, 2020)). In our typing system, $(\forall^{\text{INF}}_{\bullet\Longrightarrow})$ is such a rule: the instantiation of the quantifying variables is not known a priori. The algorithm we present in Section 4 delays this instantiation until more information about it (in particular, the set of typing constraints) is collected.

To ensure the priority of application between this rule and $(\emptyset^{\text{INF}}_{\bullet\Longrightarrow})$, we also check that the list of arguments is not empty.

Application of an arrow type Another application rule is $(\rightarrow^{\text{INF}}_{\bullet\Longrightarrow})$. This is where the subtyping is used to check that the type of the argument is convertible to (a subtype of)

$|P|$ $|N|$

$$|\alpha^+| \equiv \alpha \qquad |\alpha^-| \equiv \alpha$$
$$|{\downarrow}N| \equiv |N| \qquad |{\uparrow}P| \equiv |P|$$
$$|\exists\overrightarrow{\alpha^-}.\, P| \equiv \exists\vec{\alpha}.\, |P| \qquad |\forall\alpha^+.\, N| \equiv \forall\alpha.\, |N|$$
$$|P \to N| \equiv |P| \to |N|$$

Fig. 7: Type Depolarization [1]

$\lfloor T \rfloor$

$$\lfloor\alpha\rfloor \equiv \alpha^+$$
$$\lfloor\forall\alpha.\, T\rfloor \equiv {\downarrow}(\forall\alpha^+.\, {\uparrow}\lfloor T\rfloor)$$
$$\lfloor A \to B\rfloor \equiv {\downarrow}(\lfloor A\rfloor \to {\uparrow}\lfloor B\rfloor)$$

Fig. 8: Type Polarization

the type of the function parameter. In the algorithm (Section 4), this subtyping check will provide the constraints we need to resolve the delayed instantiations of the quantifying variables.

An important property that the declarative system has is that the declarative specification is correctly defined for equivalence classes.

Property 7 (Declarative typing is defined up to equivalence)**.** *Let us assume that* $\Theta \vdash \Gamma_1 \simeq^{\leqslant} \Gamma_2$*, i.e., the corresponding types assigned by* Γ_1 *and* Γ_2 *are equivalent in* Θ*. Also, let us assume that* $\Theta \vdash N_1 \simeq^{\leqslant} N_2$*,* $\Theta \vdash P_1 \simeq^{\leqslant} P_2$*, and* $\Theta \vdash M_1 \simeq^{\leqslant} M_2$*. Then*

- $\Theta; \Gamma_1 \vdash c : N_1$ *holds if and only if* $\Theta; \Gamma_2 \vdash c : N_2$*,*
- $\Theta; \Gamma_1 \vdash v : P_1$ *holds if and only if* $\Theta; \Gamma_2 \vdash v : P_2$*, and*
- $\Theta\,; \Gamma_1 \vdash N_1 \bullet \vec{v} \Longrightarrow M_1$ *holds if and only if* $\Theta\,; \Gamma_2 \vdash N_2 \bullet \vec{v} \Longrightarrow M_2$*.*

3.5 Relation to System F

To establish a correspondence between $\mathrm{F}^{\pm}_{\exists}$ and standard unpolarized System F, we present a translation in both ways: the polarization from System F to $\mathrm{F}^{\pm}_{\exists}$ and the depolarization from $\mathrm{F}^{\pm}_{\exists}$ to System F.

Type-level Translation The type depolarization (Fig. 27) simply forgets the polarization structure of types: it removes the shift operators and the polarities of the free type variables.

The type polarization (Fig. 28) is more involved, as there are multiple ways to polarize a type: for instance, a variable α can be polarized to α^+ or α^-. The choice of polarization affects the execution strategy of the program. In particular, a functional type can be represented in either positive (thunked, call-by-value) way: ${\downarrow}(P \to {\uparrow}Q)$ or negative (call-by-name): ${\downarrow}N \to M$. We chose the positive polarization: every System F type is translated to a *positive* type of $\mathrm{F}^{\pm}_{\exists}$. This choice makes it smoother to lift the translation to the term level, which will be discussed next.

The type-level translation is naturally lifted to the contexts: $|\Theta|$ is the context Θ with every type depolarized, and symmetrically, $\lfloor\Theta\rfloor$ is the context Θ with every type polarized. The context translation is used to formulate the soundness of the term-level translation, which we discuss next.

[1] Here the System F existential $\exists\vec{\alpha}.\, T$ is a syntax sugar for its standard encoding $\forall\beta.\, (\forall\vec{\alpha}.\, (T \to \beta)) \to \beta$

Term-level Translation The term-level translations between the systems are more complex. They are defined as *elaborations* of the typing derivations. In other words, the input of the translation is not a term, but a whole typing derivation in the corresponding system. For example, the subtyping elaboration $\Theta \vdash N \leqslant M \rightsquigarrow t$ constructs a System F function t of type $|N| \to |M|$ that witnesses the given $\mathrm{F}^{\pm}_{\exists}$-subtyping $\Theta \vdash N \leqslant M$. A typing elaboration $\Theta; \Gamma \vdash v : P \rightsquigarrow t$ builds a term t—a System F counterpart of v—of type $|P|$.

For brevity, we omit the formal definition of the term-level elaboration, referring the reader to the appendix (Section A.4.2). We define the elaboration in such a way that it preserves the soundness invariants given in Table 1.

Elaboration Judgment		The Soundness Property
Negative Subtyping	$\Theta \vdash N \leqslant M \rightsquigarrow t$	$\lvert\Theta\rvert; \cdot \vdash t : \lvert N\rvert \to \lvert M\rvert$
Positive Subtyping	$\Theta \vdash P \geqslant Q \rightsquigarrow t$	$\lvert\Theta\rvert; \cdot \vdash t : \lvert Q\rvert \to \lvert P\rvert$
Negative Typing	$\Theta; \Gamma \vdash c : N \rightsquigarrow t$	$\lvert\Theta\rvert; \lvert\Gamma\rvert \vdash t : \lvert N\rvert$
Positive Typing	$\Theta; \Gamma \vdash v : P \rightsquigarrow t$	$\lvert\Theta\rvert; \lvert\Gamma\rvert \vdash t : \lvert P\rvert$
Application Typing	$\Theta; \Gamma \vdash N \bullet \vec{v} \Longrightarrow M \rightsquigarrow e; \vec{t}$	$\lvert\Theta\rvert; \lvert\Gamma\rvert, x : \lvert N\rvert \vdash e\ (x\,\vec{t}) : \lvert M\rvert$
System F Typing	$\Theta; \Gamma \vdash t : T \rightsquigarrow^{\pm} c$	$\lfloor\Theta\rfloor; \lfloor\Gamma\rfloor \vdash c : \uparrow\lfloor T\rfloor$

Table 1: Soundness of the Elaboration: the elaboration judgment on the left implies the typing judgment on the right

The least obvious aspect of the elaboration soundness is the application typing. The elaboration judgment $\Theta; \Gamma \vdash N \bullet \vec{v} \Longrightarrow M \rightsquigarrow e; \vec{t}$ outputs two things: (i) $\vec{t}$—the System F counterpart of the argument list $\vec{v}$, and (ii) e—a System F term that is used for the final cast of the application $(((x\,t_1)\cdots)\,t_n)$ to the resulting type $|M|$. This final cast is needed to reflect the $\mathrm{F}^{\pm}_{\exists}$ feature of $(\emptyset^{\mathrm{INF}}_{\bullet\Rightarrow})$ that permits the equivalent conversion of the resulting type $\Theta \vdash N \simeq^{\leqslant} N'$, as $\mathrm{F}^{\pm}_{\exists}$-equivalent types might have different System F representations (i.e., $|N| \neq |N'|$).

4 The Algorithm

In this section, we present the algorithmization of the declarative system described above. The algorithmic system follows the structure of the declarative specification closely. First, it is also given by a set of inference rules, which, however, are mode-correct ((Dunfield and Krishnaswami, 2020)), i.e., the output of each rule is always uniquely defined by its input. And second, the rules of the algorithmic system mirrors the declarative rules (except for the rules $(\simeq^{\mathrm{INF}}_{+})$ and $(\simeq^{\mathrm{INF}}_{-})$), which simplifies the correctness proof.

4.1 Algorithmic Syntax

First, let us discuss the syntax of the algorithmic system (Fig. 9).

Algorithmic Variables To design a mode-correct inference system, we slightly modify the language we operate on. The entities (terms, types, contexts) that the algorithm manipulates we call *algorithmic*. They extend the previously defined declarative terms and types

Negative Algorithmic Variables
$\widehat{\alpha}^-, \widehat{\beta}^-, \widehat{\gamma}^-, \ldots$

Positive Algorithmic Variables
$\widehat{\alpha}^+, \widehat{\beta}^+, \widehat{\gamma}^+, \ldots$

Negative Algorithmic Types
$N, M \quad ::= \quad \cdots \quad | \quad \widehat{\alpha}^-$

Positive Algorithmic Types
$P, Q \quad ::= \quad \cdots \quad | \quad \widehat{\alpha}^+$

Algorithmic Type Context
$\widehat{\Theta} ::= \{\widehat{\alpha_1}^\pm, \ldots, \widehat{\alpha_n}^\pm\}$ where $\widehat{\alpha_1}^\pm, \ldots, \widehat{\alpha_n}^\pm$ are pairwise distinct

Instantiation Context
$\Xi ::= \{\widehat{\alpha_1}^\pm\{\Theta_1\}, \ldots, \widehat{\alpha_n}^\pm\{\Theta_n\}\}$ where $\widehat{\alpha_1}^\pm, \ldots, \widehat{\alpha_n}^\pm$ are pairwise distinct

Fig. 9: Algorithmic Syntax

by adding *algorithmic type variables* (also known as unification variables). The algorithmic variables represent unknown types, which cannot be inferred immediately but are promised to be instantiated as the algorithm proceeds.

We denote algorithmic variables as $\widehat{\alpha}^+, \widehat{\beta}^-, \ldots$ to distinguish them from normal variables α^+, β^-. In a few places, we replace the quantified variables $\overrightarrow{\alpha^+}$ with their algorithmic counterpart $\overrightarrow{\widehat{\alpha}^+}$. The procedure of replacing declarative variables with algorithmic ones we call *algorithmization* and denote as $\overrightarrow{\widehat{\alpha}^-}/\overrightarrow{\alpha^-}$ and $\overrightarrow{\widehat{\alpha}^+}/\overrightarrow{\alpha^+}$. The converse operation—*dealgorithmization*—is denoted as $\overrightarrow{\alpha^-}/\overrightarrow{\widehat{\alpha}^-}$ and is used in the least upper bound procedure (Section 4.6).

Algorithmic Types The syntax of algorithmic types extends the declarative syntax by adding algorithmic variables as new terminals. We add positive algorithmic variables $\widehat{\alpha}^+$ to the syntax of positive types, and negative algorithmic variables $\widehat{\alpha}^-$ to the syntax of negative types. All the constructors of the system can be applied to *algorithmic* types, however, algorithmic variables cannot be abstracted by the quantifiers $\forall$ and $\exists$.

Algorithmic Contexts $\widehat{\Theta}$ and Well-formedness To specify when algorithmic types are well-formed, we define algorithmic contexts $\widehat{\Theta}$ as sets of algorithmic variables. Then $\Theta\,;\widehat{\Theta} \vdash P$ and $\Theta\,;\widehat{\Theta} \vdash N$ represent the well-formedness judgment of algorithmic terms defined as expected. Informally, they check that all free declarative variables are in Θ, and all free algorithmic variables are in $\widehat{\Theta}$. Most of the rules are inherited from the well-formedness of *declarative* types: the declarative variables are checked to belong to the context Θ, the quantifiers extend the context Θ, type constructors are well-formed congruently. As we extend the syntax with algorithmic variables, we also add two base-case rules for them: ($\textsc{UVar}_+^{\textsc{WF}}$) and ($\textsc{UVar}_-^{\textsc{WF}}$) (see Fig. 10).

Instantiation Context Ξ When one instantiates an algorithmic variable, one may only use type variables *available in its scope*. As such, each algorithmic variable must remember the context at the moment when it was introduced. In our algorithm, this information is represented by an *instantiation context* Ξ—a set of pairs associating algorithmic variables and declarative contexts.

Algorithmic Substitution We define the algorithmic substitution $\widehat{\sigma}$ as a mapping from algorithmic variables to *declarative* types. The signature $\Xi \vdash \widehat{\sigma} : \widehat{\Theta}$ specifies the domain

$$\frac{\vdots}{\widehat{\alpha}^{+} \in \widehat{\Theta}} \qquad \frac{\vdots}{\widehat{\alpha}^{-} \in \widehat{\Theta}}$$
$$\frac{}{\Theta\,;\widehat{\Theta} \vdash \widehat{\alpha}^{+}}\ (\text{UVAR}_{+}^{\text{WF}}) \qquad \frac{}{\Theta\,;\widehat{\Theta} \vdash \widehat{\alpha}^{-}}\ (\text{UVAR}_{-}^{\text{WF}})$$

Fig. 10: Well-formedness of Algorithmic Types extends Declarative Well-formedness

$$\frac{\vdots}{\texttt{nf}\,(\widehat{\alpha}^{+}) = \widehat{\alpha}^{+}}\ (\text{UVAR}_{+}^{\text{NF}}) \qquad \frac{\vdots}{\texttt{nf}\,(\widehat{\alpha}^{-}) = \widehat{\alpha}^{-}}\ (\text{UVAR}_{-}^{\text{NF}})$$

Fig. 11: Normalization of Algorithmic Types extends Declarative Normalization

and the range of $\widehat{\sigma}$: for each variable $\widehat{\alpha}^{\pm}$ in $\widehat{\Theta}$, there exists an corresponding entry in Ξ associating $\widehat{\alpha}^{\pm}$ with a declarative context Θ such that $[\widehat{\sigma}]\widehat{\alpha}^{\pm}$ is well-formed in Θ. In addition, we assume that $\widehat{\sigma}$ acts as the identity on the variables not in $\widehat{\Theta}$.

Algorithmic Normalization Similarly to well-formedness, the normalization of algorithmic types is defined by extending the declarative definition (Fig. 5) with the algorithmic variables. To the rules repeating the declarative normalization, we add rules saying that normalization is trivial on algorithmic variables.

4.2 Type Constraints

Throughout the algorithm's operation, it gathers information about the algorithmic type variables, represented as *constraints*. These constraints in our system can be either *subtyping constraints* or *unification constraints*. As preserved by the algorithm, each subtyping constraint has a positive shape $\widehat{\alpha}^{+} :\geqslant P$, meaning it confines a positive algorithmic variable to be the supertype of a positive declarative type. Unification constraints can take a positive $\widehat{\alpha}^{+} :\simeq P$ or negative $\widehat{\alpha}^{-} :\simeq N$ form. Algorithmic variables cannot occur in the right-hand side of the constraints. The constraint *set* is denoted as C, and we presume that each algorithmic variable can be restricted by at most one constraint.

We define UC separately as a set solely containing *unification* constraints for simpler algorithm representation. The unification algorithm, which we use as a subroutine of the subtyping algorithm, can only produce unification constraints. The resolution of unification constraints is simpler than that of a general constraint set. This way, this separation allows us to better decompose the algorithm's structure, thus simplifying the inductive proofs.

Auxiliary Functions

- We define $\texttt{dom}\,(C)$—the domain of constraint set C—as a set of algorithmic variables that it restricts. Similarly, we define $\texttt{dom}\,(\Xi)$—the domain of instantiation context Ξ—as a set of algorithmic variables that Ξ associates with their contexts.
- We write $\Xi(\widehat{\alpha}^{\pm})$ to denote the declarative context associated with $\widehat{\alpha}^{\pm}$ in Ξ.

Constraint Entry		Unification Constraint Entry	
e	$::=$	ue	$::=$
	$\widehat{\alpha}^{+} :\simeqslant P$		$\widehat{\alpha}^{+} :\simeq P$
	$\widehat{\alpha}^{-} :\simeqslant N$		$\widehat{\alpha}^{-} :\simeq N$
Constraint Set		Unification Constraint Set	
C	$::= \{e_1, \ldots, e_n\}$	UC	$::= \{ue_1, \ldots, ue_n\}$

Fig. 12: Constraint Entries and Sets

- $\mathtt{fav}(N)$ and $\mathtt{fav}(P)$ denote the set of free algorithmic variables in N and P respectively.
- We write $\Theta \vdash^{\supseteq} \Xi$ to denote that each declarative context associated with an algorithmic variable in Ξ is a subcontext (subset) of Θ.

Equivalent Substitutions In the proofs of the algorithm correctness, we often state or require that two substitutions are equivalent on a given set of algorithmic variables. We denote it as $\Xi \vdash \widehat{\sigma}' \simeq^{\leqslant} \widehat{\sigma} : \widehat{\Theta}$ meaning that for each $\widehat{\alpha}^{\pm}$ in $\widehat{\Theta}$, substitutions $\widehat{\sigma}$ and $\widehat{\sigma}'$ map $\widehat{\alpha}^{\pm}$ to types equivalent in the corresponding context: $\Xi(\widehat{\alpha}^{\pm}) \vdash [\widehat{\sigma}']\widehat{\alpha}^{\pm} \simeq^{\leqslant} [\widehat{\sigma}]\widehat{\alpha}^{\pm}$.

Constraint well-formedness and satisfaction Suppose that Ξ is an instantiation context. We say that constraint set C is well-formed in Ξ (denoted as $\Xi \vdash C$) if for every constrain entry $e \in C$ associating a variable $\widehat{\alpha}^{\pm}$ with a type P, this type is well-formed in the corresponding declarative context $\Xi(\widehat{\alpha}^{\pm})$.

Substitution $\widehat{\sigma}$ satisfies a constraint *entry* e restricting algorithmic variable $\widehat{\alpha}^{\pm}$ if $[\sigma]\widehat{\alpha}^{\pm}$ can be validly substituted for $\widehat{\alpha}^{\pm}$ in e (so that the corresponding equivalence or subtyping holds).

Substitution $\widehat{\sigma}$ satisfies a constraint *set* C if $\Xi \vdash C$ and *each entry* of C is satisfied by $\widehat{\sigma}$.

4.3 Subtyping Algorithm

We decompose the subtyping algorithm into several procedures. Figure 13 shows these procedures and the dependencies between them: arrows denote the invocation of one procedure from another.

Some of the procedures (in particular, the unification and the anti-unification) assume that the input types are normalized. Therefore, we call the normalization procedure before invoking them, indicating this by the '$\mathtt{nf}$' annotation on the arrows in Fig. 13. Alternatively, one could normalize the input types in the very beginning and preserve the normalization throughout the algorithm. However, we delay the normalization to the places where it is required to show that normalization is needed only at these stages to maintain consistent invariants.

In the remainder of this section, we will discuss each of these procedures in detail, following the top-down order of the dependency graph. First, we present the subtyping algorithm itself.

As an input, the subtyping algorithm takes a type context Θ, an instantiation context Ξ, and two types of the corresponding polarity: N and M for the negative subtyping, and P and Q for the positive subtyping. We assume the second type (M and Q) to be declarative (with

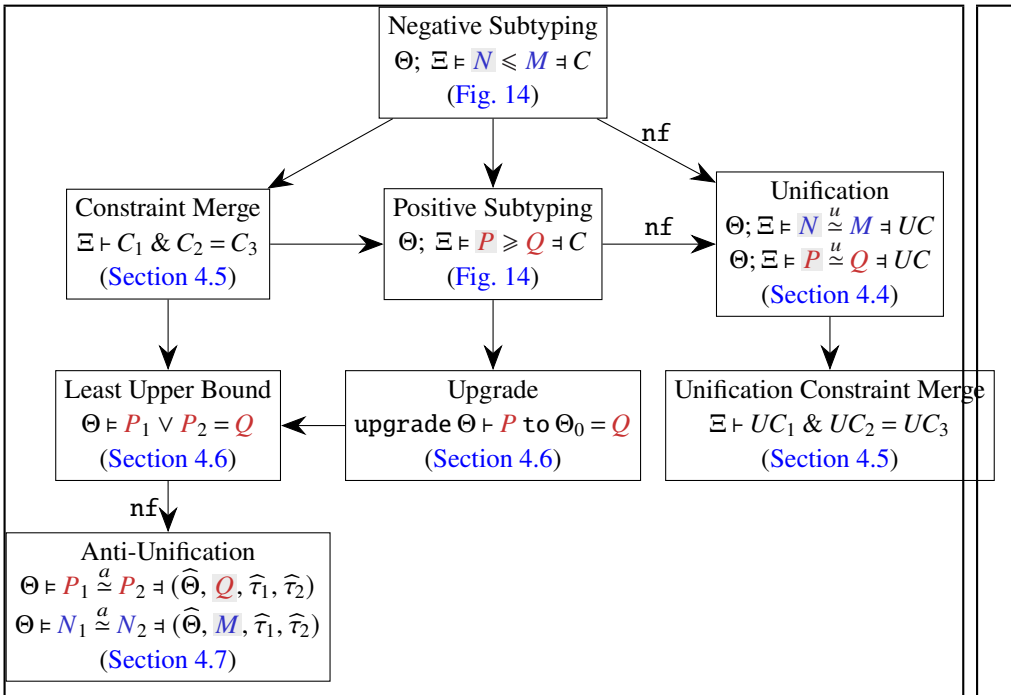

Fig. 13: Dependency graph of the subtyping algorithm

no algorithmic variables) and well-formed in Θ, but the first type (N and P) may contain algorithmic variables, whose instantiation contexts are specified by Ξ.

Notice that the shape of the input types uniquely determines the applied subtyping rule. If the subtyping is successful, it returns a set of constraints C restricting the algorithmic variables of the first type. If the subtyping does not hold, there will be no inference tree with such inputs.

The rules of the subtyping algorithm bijectively correspond to the rules of the declarative system. Let us discuss them in detail.

Variables Rules ($\textsc{Var}^{\leqslant}_{-}$) and ($\textsc{Var}^{\geqslant}_{+}$) say that if both of the input types are equal declarative variables, they are subtypes of each other, with no constraints (as there are no algorithmic variables).

Shifts Rules ($\downarrow^{\geqslant}$) and ($\uparrow^{\leqslant}$) cover the downshift and the upshift cases, respectively. If the input types are constructed by shifts, then the subtyping can only hold if they are equivalent. This way, the algorithm must find the instantiations of the algorithmic variables on the left-hand side such that these instantiations make the left-hand side and the ride-hand side equivalent. For this purpose, the algorithm invokes the unification procedure (Section 4.4) preceded by the normalization of the input types. It returns the resulting constraints given by the unification algorithm.

Quantifiers Rules ($\forall^{\leqslant}$) and ($\exists^{\geqslant}$) are symmetric. Declaratively, the quantified variables on the left-hand side must be instantiated with types, which are not known beforehand. We address this problem by algorithmization of the quantified variables (see Section 4.1). The rule introduces fresh algorithmic variables $\overrightarrow{\widehat{\alpha}^{+}}$ or $\overrightarrow{\widehat{\alpha}^{-}}$, puts them into the instantiation

$\boxed{\Theta;\ \Xi \vDash N \leqslant M \dashv C}$ Negative subtyping $\qquad$ $\boxed{\Theta;\ \Xi \vDash P \geqslant Q \dashv C}$ Positive supertyping

$$\frac{}{\Theta;\ \Xi \vDash \alpha^- \leqslant \alpha^- \dashv \cdot}\ (\textsc{Var}^{\leqslant}_{-}) \qquad \frac{}{\Theta;\ \Xi \vDash \alpha^+ \geqslant \alpha^+ \dashv \cdot}\ (\textsc{Var}^{\geqslant}_{+})$$

$$\frac{\Theta;\ \Xi \vDash \mathtt{nf}\,(P) \overset{u}{\simeq} \mathtt{nf}\,(Q) \dashv UC}{\Theta;\ \Xi \vDash \uparrow P \leqslant \uparrow Q \dashv UC}\ (\uparrow^{\leqslant}) \qquad \frac{\overrightarrow{\widehat{\alpha}^-}\ \mathtt{are\ fresh} \quad \Theta, \overrightarrow{\beta^-};\ \Xi, \overrightarrow{\widehat{\alpha}^-}\{\Theta, \overrightarrow{\beta^-}\} \vDash [\overrightarrow{\widehat{\alpha}^-}/\overrightarrow{\alpha^-}]P \geqslant Q \dashv C}{\Theta;\ \Xi \vDash \exists \overrightarrow{\alpha^-}.\ P \geqslant \exists \overrightarrow{\beta^-}.\ Q \dashv C \setminus \overrightarrow{\widehat{\alpha}^-}}\ (\exists^{\geqslant})$$

$$\frac{\overrightarrow{\widehat{\alpha}^+}\ \mathtt{are\ fresh} \quad \Theta, \overrightarrow{\beta^+};\ \Xi, \overrightarrow{\widehat{\alpha}^+}\{\Theta, \overrightarrow{\beta^+}\} \vDash [\overrightarrow{\widehat{\alpha}^+}/\overrightarrow{\alpha^+}]N \leqslant M \dashv C}{\Theta;\ \Xi \vDash \forall \overrightarrow{\alpha^+}.\ N \leqslant \forall \overrightarrow{\beta^+}.\ M \dashv C \setminus \overrightarrow{\widehat{\alpha}^+}}\ (\forall^{\leqslant}) \qquad \frac{\Theta;\ \Xi \vDash \mathtt{nf}\,(N) \overset{u}{\simeq} \mathtt{nf}\,(M) \dashv UC}{\Theta;\ \Xi \vDash \downarrow N \geqslant \downarrow M \dashv UC}\ (\downarrow^{\geqslant})$$

$$\frac{\Theta;\ \Xi \vDash P \geqslant Q \dashv C_1 \quad \Theta;\ \Xi \vDash N \leqslant M \dashv C_2 \quad \Xi \vdash C_1 \,\&\, C_2 = C}{\Theta;\ \Xi \vDash P \to N \leqslant Q \to M \dashv C}\ (\to^{\leqslant}) \qquad \frac{\mathtt{upgrade}\ \Theta \vdash P\ \mathtt{to}\ \Xi(\widehat{\alpha}^+) = Q}{\Theta;\ \Xi \vDash \widehat{\alpha}^+ \geqslant P \dashv (\widehat{\alpha}^+ :\geqslant Q)}\ (\textsc{UVar}^{\geqslant})$$

Fig. 14: Subtyping Algorithm

context Ξ (specifying that they must be instantiated in the extended context $\Theta, \overrightarrow{\beta^+}$ or $\Theta, \overrightarrow{\beta^-}$) and substitute the quantified variables for them in the input type.

After algorithmization of the quantified variables, the algorithm proceeds with the recursive call, returning constraints C. As the output, the algorithm removes the freshly introduced algorithmic variables from the instantiation context, This operation is sound: it is guaranteed that C always has a solution, but the specific instantiation of the freshly introduced algorithmic variables is not important, as they do not occur in the input types.

Functions To infer the subtyping of the function types, the algorithm makes two calls: (i) a recursive call ensuring the subtyping of the result types, and (ii) a call to positive subtyping (or rather super-typing) on the argument types. The resulting constraints are merged using a special procedure defined in Section 4.5 and returned as the output.

Algorithmic variables If one of the sides of the subtyping is a unification variable, the algorithm creates a new constraint. Because the right-hand side of the subtyping is always declarative, it is only the left-hand side that can be a unification variable. Moreover, another invariant we preserve prevents the negative algorithmic variables from occurring in types during the negative subtyping algorithm. It means that the only possible form of the subtyping here is $\widehat{\alpha}^+ \geqslant P$, which is covered by $(\textsc{UVar}^{\geqslant})$.

The potential problem here is that the type P might be not well-formed in the instantiation context required for $\widehat{\alpha}^+$ by Ξ because this context might be smaller than the current context Θ. As we wish the resulting constraint set to be sound w.r.t. Ξ, we cannot simply put $\widehat{\alpha}^+ :\geqslant P$ into the output. Prior to that, we update the type P to its lowest supertype Q well-formed in $\Xi(\widehat{\alpha}^+)$. It is done by the *upgrade* procedure, which we discuss in detail in Section 4.6.

To summarize, the subtyping algorithm uses the following additional subroutines: (i) rules $(\downarrow^{\geqslant})$ and $(\uparrow^{\leqslant})$ invoke the *unification* algorithm to equate the input types; (ii) rule $(\to^{\leqslant})$

$\boxed{\Theta; \Xi \vDash N \overset{u}{\simeq} M \dashv UC}$ Negative unification $\qquad \boxed{\Theta; \Xi \vDash P \overset{u}{\simeq} Q \dashv UC}$ Positive unification

$$\frac{}{\Theta; \Xi \vDash \alpha^- \overset{u}{\simeq} \alpha^- \dashv \cdot}\ (\textsc{Var}^{\overset{u}{\simeq}}_{-}) \qquad \frac{}{\Theta; \Xi \vDash \alpha^+ \overset{u}{\simeq} \alpha^+ \dashv \cdot}\ (\textsc{Var}^{\overset{u}{\simeq}}_{+})$$

$$\frac{\Theta; \Xi \vDash P \overset{u}{\simeq} Q \dashv UC}{\Theta; \Xi \vDash \uparrow P \overset{u}{\simeq} \uparrow Q \dashv UC}\ (\uparrow^{\overset{u}{\simeq}}) \qquad \frac{\Theta; \Xi \vDash N \overset{u}{\simeq} M \dashv UC}{\Theta; \Xi \vDash \downarrow N \overset{u}{\simeq} \downarrow M \dashv UC}\ (\downarrow^{\overset{u}{\simeq}})$$

$$\frac{\Theta; \Xi \vDash P \overset{u}{\simeq} Q \dashv UC_1 \quad \Theta; \Xi \vDash N \overset{u}{\simeq} M \dashv UC_2}{\Theta; \Xi \vDash P \to N \overset{u}{\simeq} Q \to M \dashv UC_1 \,\&\, UC_2}\ (\to^{\overset{u}{\simeq}}) \qquad \frac{\Theta, \overrightarrow{\alpha^-}; \Xi \vDash P \overset{u}{\simeq} Q \dashv UC}{\Theta; \Xi \vDash \exists \overrightarrow{\alpha^-}.\, P \overset{u}{\simeq} \exists \overrightarrow{\alpha^-}.\, Q \dashv UC}\ (\exists^{\overset{u}{\simeq}})$$

$$\frac{\Theta, \overrightarrow{\alpha^+}; \Xi \vDash N \overset{u}{\simeq} M \dashv UC}{\Theta; \Xi \vDash \forall \overrightarrow{\alpha^+}.\, N \overset{u}{\simeq} \forall \overrightarrow{\alpha^+}.\, M \dashv UC}\ (\forall^{\overset{u}{\simeq}}) \qquad \frac{\Xi(\widehat{\alpha}^+) \vdash P}{\Theta; \Xi \vDash \widehat{\alpha}^+ \overset{u}{\simeq} P \dashv (\widehat{\alpha}^+ :\simeq P)}\ (\textsc{UVar}^{\overset{u}{\simeq}}_{+})$$

$$\frac{\Xi(\widehat{\alpha}^-) \vdash N}{\Theta; \Xi \vDash \widehat{\alpha}^- \overset{u}{\simeq} N \dashv (\widehat{\alpha}^- :\simeq N)}\ (\textsc{UVar}^{\overset{u}{\simeq}}_{-})$$

Fig. 15: Unification Algorithm

merges the constraints produced by the recursive calls on the result and the argument types; and (iii) rule ($\textsc{UVar}^{\geqslant}$) *upgrades* the input type to its least supertype well-formed in the context required by the algorithmic variable. The following sections discuss these additional procedures in detail.

4.4 Unification

As an input, the unification context takes a type context Θ, an instantiation context Ξ, and two types of the required polarity: N and M for the negative unification, and P and Q for the positive unification. It is assumed that only the left-hand side type may contain algorithmic variables. This way, the left-hand side is well-formed as an algorithmic type in Θ and Ξ, whereas the right-hand side is well-formed declaratively in Θ.

Since only the left-hand side may contain algorithmic variables that the unification instantiates, we could have called this procedure *matching*. However, in Section 6.2, we will discuss several modifications of the type system, where this invariant is not preserved, and therefore, this procedure requires general first-order pattern unification (Miller, 1991).

As the output, the unification algorithm returns *the weakest* set of unification constraints UC such that *any* instantiation satisfying these constraints unifies the input types.

The algorithm works as one might expect: if both sides are formed by constructors, it is required that the constructors are the same, and the types unify recursively. If one of the sides is a unification variable (in our case it can only be the left-hand side), we create a new unification constraint restricting it to be equal to the other side. Let us discuss the rules that implement this strategy.

Variables The variable rules ($\textsc{Var}^{\overset{u}{\simeq}}_{-}$) and ($\textsc{Var}^{\overset{u}{\simeq}}_{+}$) are trivial: as the input types do not have algorithmic variables, and are already equal, the unification returns no constraints.

Shifts The shift rules ($\downarrow^{\overset{u}{\simeq}}$) and ($\uparrow^{\overset{u}{\simeq}}$) require the two input types to be formed by the same shift constructor. They remove this constructor, unify the types recursively, and return the resulting set of constraints.

Quantifiers Similarly, the quantifier rules ($\forall^{\overset{u}{\simeq}}$) and ($\exists^{\overset{u}{\simeq}}$) require the quantifier variables on the left-hand side and the right-hand side to be the same. This requirement is complete because we assume the input types of the unification to be normalized, and thus, the equivalence implies alpha-equivalence. In the implementation of this rule, an alpha-renaming might be needed to ensure that the quantified variables are the same, however, we omit it for brevity.

Functions Rule ($\rightarrow^{\overset{u}{\simeq}}$) unifies two functional types. First, it unifies their argument types and their result types recursively. Then it merges the resulting constraints using the procedure described in Section 4.5.

Notice that the resulting constraints can only have *unification* entries. It means that they can be merged in a simpler way than general constraints. In particular, the merging procedure does not call any of the subtyping subroutines but rather simply checks the matching constraint entries for equality.

Algorithmic variable Finally, if the left-hand side of the unification is an algorithmic variable, ($\textsc{Var}^{\overset{u}{\simeq}}_{-}$) or ($\textsc{Var}^{\overset{u}{\simeq}}_{+}$) is applied. It simply checks that the right-hand side type is well-formed in the required instantiation context, and returns a newly created constraint restricting the variable to be equal to the right-hand side type.

As one can see, the unification procedure is standard; the only peculiarity (although it is common for type inference) is that it makes sure that the resulting instantiations agree with the input instantiation context Ξ. As a subroutine, the unification algorithm only uses the (unification) constraint merge procedure and the well-formedness checking.

4.5 Constraint Merge

In this section, we discuss the constraint merging procedure. It allows one to combine two constraint sets into one. A simple union of two constraint sets is not sufficient, since the resulting set must not contain two entries restricting the same algorithmic variable—we call such entries *matching*.

The matching entries must be combined into *one* constraint entry, that would represent their conjunction. This way, to merge two constraint sets, we unite the entries of two sets and then merge the matching pairs.

Merging matching constraint entries Two *matching* entries formed in the same context Θ can be merged as shown in Fig. 16. Suppose that e_1 and e_2 are input entries. The result of the merge e_1 & e_2 is *the weakest entry which implies both* e_1 *and* e_2.

Suppose that one of the input entries, say e_1, is a *unification* constraint entry. Then the resulting entry e_1 must coincide with it (up-to-equivalence), and thus, it is only required to check that e_2 is implied by e_1. We consider two options:

$\Theta \vdash e_1 \,\&\, e_2 = e_3$ Subtyping Constraint Entry Merge

$$\frac{\Theta \vDash P_1 \vee P_2 = Q}{\Theta \vdash (\widehat{\alpha}^+ :\geqslant P_1) \,\&\, (\widehat{\alpha}^+ :\geqslant P_2) = (\widehat{\alpha}^+ :\geqslant Q)} \; (\geqslant \&^+ \geqslant)$$

$$\frac{\Theta;\, \cdot \vDash P \geqslant Q \dashv \cdot}{\Theta \vdash (\widehat{\alpha}^+ :\simeq P) \,\&\, (\widehat{\alpha}^+ :\geqslant Q) = (\widehat{\alpha}^+ :\simeq P)} \; (\simeq \&^+ \geqslant)$$

$$\frac{\Theta;\, \cdot \vDash Q \geqslant P \dashv \cdot}{\Theta \vdash (\widehat{\alpha}^+ :\geqslant P) \,\&\, (\widehat{\alpha}^+ :\simeq Q) = (\widehat{\alpha}^+ :\simeq Q)} \; (\geqslant \&^+ \simeq)$$

$$\frac{\mathtt{nf}\,(P) = \mathtt{nf}\,(P')}{\Theta \vdash (\widehat{\alpha}^+ :\simeq P) \,\&\, (\widehat{\alpha}^+ :\simeq P') = (\widehat{\alpha}^+ :\simeq P)} \; (\simeq \&^+ \simeq)$$

$$\frac{\mathtt{nf}\,(N) = \mathtt{nf}\,(N')}{\Theta \vdash (\widehat{\alpha}^- :\simeq N) \,\&\, (\widehat{\alpha}^- :\simeq N') = (\widehat{\alpha}^- :\simeq N)} \; (\simeq \&^- \simeq)$$

Fig. 16: Merge of Matching Constraint Entries

Suppose that $\Xi \vdash C_1$ and $\Xi \vdash C_2$.
Then $\Xi \vdash C_1 \,\&\, C_2 = C$ defines a set of constraints C such that $e \in C$ iff either:

- $e \in C_1$ and there is no matching $e' \in C_2$; or
- $e \in C_2$ and there is no matching $e' \in C_1$; or
- $\Xi(\widehat{\alpha}^{\pm}) \vdash e_1 \,\&\, e_2 = e$ for some $e_1 \in C_1$ and $e_2 \in C_2$ such that e_1 and e_2 both restrict variable $\widehat{\alpha}^{\pm}$.

Fig. 17: Constraint Merge

(i) if e_2 is also a *unification* entry, then the types on the right-hand side of e_1 and e_2 must be equivalent, as given by rules $(\simeq \&^+ \simeq)$ and $(\simeq \&^- \simeq)$;

(ii) if e_2 is a *supertype* constraint entry $\widehat{\alpha}^+ :\geqslant P$, the algorithm must check that the type assigned by e_1 is a supertype of P. The corresponding symmetric rules are $(\geqslant \&^+ \simeq)$ and $(\simeq \&^+ \geqslant)$. In the premises of these rules, P is the same as P below, and Q is the same as Q below but relaxed to an algorithmic type.

If both input entries are supertype constraints: $\widehat{\alpha}^+ :\geqslant P$ and $\widehat{\alpha}^+ :\geqslant Q$, then their conjunction is $\widehat{\alpha}^+ :\geqslant P \vee Q$, as given by $(\geqslant \&^+ \geqslant)$. The least upper bound—$P \vee Q$—is the least supertype of both P and Q, and this way, $\widehat{\alpha}^+ :\geqslant P \vee Q$ is the weakest constraint entry that implies $\widehat{\alpha}^+ :\geqslant P$ and $\widehat{\alpha}^+ :\geqslant Q$. The algorithm finding the least upper bound is discussed in Section 4.6.

Merging constraint sets The algorithm for merging constraint sets is shown in Fig. 17. As discussed, the result of merge C_1 and C_2 consists of three parts: (i) the entries of C_1 that do not match any entry of C_2; (ii) the entries of C_2 that do not match any entry of C_1; and (iii) the merge (Fig. 16) of matching entries.

As shown in Fig. 16, the merging procedure relies substantially on the least upper bound algorithm. In the next section, we discuss this algorithm in detail, together with the upgrade procedure, selecting the least supertype well-formed *in a given context*.

$\boxed{\texttt{upgrade}\ \Theta \vdash P\ \texttt{to}\ \Theta_0 = Q}$ Type Upgrade

$$\frac{\begin{array}{l}\Theta = \Theta_0, \overrightarrow{\alpha^{\pm}} \\ \overrightarrow{\beta^{\pm}}\ \texttt{are fresh} \quad \overrightarrow{\gamma^{\pm}}\ \texttt{are fresh} \\ \Theta_0, \overrightarrow{\beta^{\pm}}, \overrightarrow{\gamma^{\pm}} \vDash [\overrightarrow{\beta^{\pm}}/\overrightarrow{\alpha^{\pm}}]P \vee [\overrightarrow{\gamma^{\pm}}/\overrightarrow{\alpha^{\pm}}]P = Q\end{array}}{\texttt{upgrade}\ \Theta \vdash P\ \texttt{to}\ \Theta_0 = Q}\ (\textsc{Upg})$$

$\boxed{\Theta \vDash P_1 \vee P_2 = Q}$ Least Upper Bound

$$\frac{\Theta, \overrightarrow{\alpha^{-}}, \overrightarrow{\beta^{-}} \vDash P_1 \vee P_2 = Q}{\Theta \vDash \exists\overrightarrow{\alpha^{-}}.\ P_1 \vee \exists\overrightarrow{\beta^{-}}.\ P_2 = Q}\ (\exists^{\vee})$$

$$\frac{}{\Theta \vDash \alpha^{+} \vee \alpha^{+} = \alpha^{+}}\ (\textsc{Var}^{\vee})$$

$$\frac{\Theta \vDash \texttt{nf}\ (\downarrow N) \overset{a}{\simeq} \texttt{nf}\ (\downarrow M) \dashv (\widehat{\Theta}, P, \widehat{\tau_1}, \widehat{\tau_2})}{\Theta \vDash \downarrow N \vee \downarrow M = \exists\overrightarrow{\alpha^{-}}.\ [\overrightarrow{\alpha^{-}}/\widehat{\Theta}]P}\ (\downarrow^{\vee})$$

Fig. 18: Type Upgrade and Leas Upper Bound Algorithms

4.6 Type Upgrade and the Least Upper Bounds

Both type upgrade and the least upper bound algorithms are used to find a minimal supertype under certain conditions. For a given type P well-formed in Θ, the *upgrade* operation finds the least among those supertypes of P that are well-formed in a smaller context $\Theta_0 \subseteq \Theta$. When given two types P_1 and P_2 well-formed in Θ, the *least upper bound* operation finds the least among common supertypes of P_1 and P_2 well-formed in Θ. These algorithms are shown in Fig. 18.

The Type Upgrade The type upgrade algorithm uses the least upper bound algorithm as a subroutine. It exploits the idea that the free variables of a positive type Q cannot disappear in its subtypes (see Property 1). It means that if a type P has free variables not occurring in P', then any common supertype of P and P' must not contain these variables either. This way, any supertype of P not containing certain variables $\overrightarrow{\alpha^{\pm}}$ must also be a supertype of $P' = [\overrightarrow{\beta^{\pm}}/\overrightarrow{\alpha^{\pm}}]P$, where $\overrightarrow{\beta^{\pm}}$ are fresh; and vice versa: any common supertype of P and P' does not contain $\overrightarrow{\alpha^{\pm}}$ or $\overrightarrow{\beta^{\pm}}$.

This way, to find the least supertype of P well-formed in $\Theta_0 = \Theta \setminus \overrightarrow{\alpha^{\pm}}$ (i.e., not containing $\overrightarrow{\alpha^{\pm}}$), we can do the following. First, construct a new type P' by renaming $\overrightarrow{\alpha^{\pm}}$ in P to fresh $\overrightarrow{\beta^{\pm}}$, and second, find *the least upper bound* of P and P' in the appropriate context. However, for reasons of symmetry, in rule (Upg) we employ a different but equivalent approach: we create *two* types P_1 and P_2 by renaming $\overrightarrow{\alpha^{\pm}}$ in P to fresh disjoint variables $\overrightarrow{\beta^{\pm}}$ and $\overrightarrow{\gamma^{\pm}}$ respectively, and then find the least upper bound of P_1 and P_2.

The Least Upper Bound The Least Upper Bound algorithm we use operates on *positive* types. This way, the inference rules of the algorithm analyze the three possible shapes of the input types: a variable type, an existential type, and a shifted computation.

Rule ($\exists^{\vee}$) covers the case when at least one of the input types is an existential type. In this case, we can simply move the existential quantifiers from both sides to the context, and make a tail-recursive call. However, it is important to make sure that the quantified variables $\overrightarrow{\alpha^{-}}$ and $\overrightarrow{\beta^{-}}$ are disjoint (i.e., alpha-renaming might be required in the implementation).

Rule ($\textsc{Var}^{\vee}$) applies when both sides are variables. In this case, the common supertype only exists if these variables are the same. And if they are, the common supertypes must be equivalent to this variable.

Rule $(\downarrow^{\vee})$ is the most interesting. If both sides are not quantified, and one of the sides is a shift, so must be the other side. However, the set of common upper bounds is not trivial in this case. For example, $\downarrow(\beta^+ \to \gamma_1^-)$ and $\downarrow(\beta^+ \to \gamma_2^-)$ have two non-equivalent common supertypes: $\exists\alpha^-.\downarrow\alpha^-$ (by instantiating α^- with $\beta^+ \to \gamma_1^-$ and $\beta^+ \to \gamma_2^-$ respectively) and $\exists\alpha^-.\downarrow(\beta^+ \to \alpha^-)$ (by instantiating α^- with γ_1^- and γ_2^- respectively). As one can see, the second supertype $\exists\alpha^-.\downarrow(\beta^+ \to \alpha^-)$ is the least among them because it abstracts over a 'deeper' negative subexpression.

In general, we must (i) find the most detailed pattern (a type with 'holes' at negative positions) that matches both sides, and (ii) abstract over the 'holes' by existential quantifiers. The algorithm that finds the most detailed common pattern is called *anti-unification*. As output, it returns $(\widehat{\Theta}, P, \widehat{\tau}_1, \widehat{\tau}_2)$, where important for us is P—the pattern—and $\widehat{\Theta}$–the set of 'holes' represented by negative algorithmic variables. We discuss the anti-unification algorithm in detail in the following section.

4.7 Anti-Unification

The anti-unification algorithm (Plotkin, 1970; Reynolds, 1970), is a procedure dual to unification. For two given (potentially different) expressions, it finds the most specific generalizer—the most detailed pattern that matches both of the input expressions. As evidence, it can also return two substitutions that instantiate the 'holes' of the pattern to the input expressions.

In our case, we have to be more demanding on the anti-unification algorithm. Since we use it to construct an existential type, whose (negative) quantified variables can only be instantiated with negative types, we must make sure that the pattern has 'holes' only at negative positions. Moreover, we must make sure that the resulting substitutions for the 'holes' are well-formed in the context from the past—at the moment when the corresponding polymorphic variables were introduced—and do not contain variables bound later. For example, the anti-unification of $N_1 = \forall\beta^+.\,\alpha_1^+ \to \uparrow\beta^+$ and $N_2 = \forall\beta^+.\,\alpha_2^+ \to \uparrow\beta^+$ is a singleton 'hole', which we model as an algorithmic type variable $\widehat{\gamma}^-$, with a pair of substitutions $\widehat{\gamma}^- \mapsto N_1$ and $\widehat{\gamma}^- \mapsto N_2$. But it *cannot* be more specific such as $\forall\beta^+.\,\widehat{\gamma}^+ \to \uparrow\beta^+$ (since the hole cannot be positive) or $\forall\beta^+.\,\widehat{\gamma}^-$ (since *the instantiation of* $\widehat{\gamma}^-$ *cannot capture the bound variable* β^+).

The algorithm that finds the most specific generalizer of two types under required conditions is given in Fig. 19. It consists of two mutually recursive procedures: the positive and the negative anti-unification. As the positive and the negative anti-unification procedures are symmetric in their interface, let us discuss how to read the positive judgment.

The positive anti-unification judgment has form $\Theta \vDash P_1 \overset{a}{\simeq} P_2 \dashv (\widehat{\Theta}, Q, \widehat{\tau}_1, \widehat{\tau}_2)$. As an input, it takes a context Θ, in which the 'holes' instantiations must be well-formed, and two positive types: P_1 and P_2; it returns a tuple of four components: $\widehat{\Theta}$—a set of 'holes' represented by negative algorithmic variables, Q—a pattern represented as a positive algorithmic type, whose algorithmic variables are in $\widehat{\Theta}$, and two substitutions $\widehat{\tau}_1$ and $\widehat{\tau}_2$ instantiating the variables from $\widehat{\Theta}$ such that $[\widehat{\tau}_1]\,Q = P_1$ and $[\widehat{\tau}_2]\,Q = P_2$.

$\boxed{\Theta \vDash P_1 \overset{a}{\simeq} P_2 \dashv (\widehat{\Theta}, Q, \widehat{\tau}_1, \widehat{\tau}_2)}$

$$\frac{}{\Theta \vDash \alpha^+ \overset{a}{\simeq} \alpha^+ \dashv (\cdot, \alpha^+, \cdot, \cdot)}\ (\textsc{Var}^{\overset{a}{\simeq}}_+)$$

$$\frac{\Theta \vDash N_1 \overset{a}{\simeq} N_2 \dashv (\widehat{\Theta}, M, \widehat{\tau}_1, \widehat{\tau}_2)}{\Theta \vDash \downarrow N_1 \overset{a}{\simeq} \downarrow N_2 \dashv (\widehat{\Theta}, \downarrow M, \widehat{\tau}_1, \widehat{\tau}_2)}\ (\downarrow^{\overset{a}{\simeq}})$$

$$\frac{\overrightarrow{\alpha^-} \cap \Theta = \emptyset \quad \Theta \vDash P_1 \overset{a}{\simeq} P_2 \dashv (\widehat{\Theta}, Q, \widehat{\tau}_1, \widehat{\tau}_2)}{\Theta \vDash \exists \overrightarrow{\alpha^-}.\ P_1 \overset{a}{\simeq} \exists \overrightarrow{\alpha^-}.\ P_2 \dashv (\widehat{\Theta}, \exists \overrightarrow{\alpha^-}.\ Q, \widehat{\tau}_1, \widehat{\tau}_2)}\ (\exists^{\overset{a}{\simeq}})$$

$\boxed{\Theta \vDash N_1 \overset{a}{\simeq} N_2 \dashv (\widehat{\Theta}, M, \widehat{\tau}_1, \widehat{\tau}_2)}$

$$\frac{}{\Theta \vDash \alpha^- \overset{a}{\simeq} \alpha^- \dashv (\cdot, \alpha^-, \cdot, \cdot)}\ (\textsc{Var}^{\overset{a}{\simeq}}_-)$$

$$\frac{\Theta \vDash P_1 \overset{a}{\simeq} P_2 \dashv (\widehat{\Theta}, Q, \widehat{\tau}_1, \widehat{\tau}_2)}{\Theta \vDash \uparrow P_1 \overset{a}{\simeq} \uparrow P_2 \dashv (\widehat{\Theta}, \uparrow Q, \widehat{\tau}_1, \widehat{\tau}_2)}\ (\uparrow^{\overset{a}{\simeq}})$$

$$\frac{\overrightarrow{\alpha^+} \cap \Theta = \emptyset \quad \Theta \vDash N_1 \overset{a}{\simeq} N_2 \dashv (\widehat{\Theta}, M, \widehat{\tau}_1, \widehat{\tau}_2)}{\Theta \vDash \forall \overrightarrow{\alpha^+}.\ N_1 \overset{a}{\simeq} \forall \overrightarrow{\alpha^+}.\ N_2 \dashv (\widehat{\Theta}, \forall \overrightarrow{\alpha^+}.\ M, \widehat{\tau}_1, \widehat{\tau}_2)}\ (\forall^{\overset{a}{\simeq}})$$

$$\frac{\Theta \vDash P_1 \overset{a}{\simeq} P_2 \dashv (\widehat{\Theta}_1, Q, \widehat{\tau}_1, \widehat{\tau}_2) \quad \Theta \vDash N_1 \overset{a}{\simeq} N_2 \dashv (\widehat{\Theta}_2, M, \widehat{\tau}'_1, \widehat{\tau}'_2)}{\Theta \vDash P_1 \to N_1 \overset{a}{\simeq} P_2 \to N_2 \dashv (\widehat{\Theta}_1 \cup \widehat{\Theta}_2, Q \to M, \widehat{\tau}_1 \cup \widehat{\tau}'_1, \widehat{\tau}_2 \cup \widehat{\tau}'_2)}\ (\to^{\overset{a}{\simeq}})$$

$$\frac{\text{if other rules are not applicable} \quad \Theta \vdash N \quad \Theta \vdash M}{\Theta \vDash N \overset{a}{\simeq} M \dashv (\widehat{\alpha}^-_{\{N,M\}}, \widehat{\alpha}^-_{\{N,M\}}, (\widehat{\alpha}^-_{\{N,M\}} \mapsto N), (\widehat{\alpha}^-_{\{N,M\}} \mapsto M))}\ (\text{AU})$$

Fig. 19: Anti-Unification Algorithm

At the high level, the algorithm scheme follows the standard approach (Plotkin, 1970) represented as a recursive procedure. Specifically, it follows two principles:

(i) if the input terms start with the same constructor, we anti-unify the corresponding parts recursively and unite the results. This principle is followed by all the rules except (AU), which works as follows:

(ii) if the first principle does not apply to the input terms N and M (for instance, if they have different outer constructors), the anti-unification algorithm returns a 'hole' such that one substitution maps it to N and the other maps it to M. This 'hole' should have a name uniquely defined by the pair (N, M), so that it automatically merges with other 'holes' mapped to the same pair of types, and thus, the initiality of the generalizer is ensured.

Let us discuss the specific rules of the algorithm in detail.

Variables Rules $(\textsc{Var}^{\overset{a}{\simeq}}_+)$ and $(\textsc{Var}^{\overset{a}{\simeq}}_-)$ generalize two equal variables. In this case, the resulting pattern is the variable itself, and no 'holes' are needed.

Shifts Rules $(\downarrow^{\overset{a}{\simeq}})$ and $(\uparrow^{\overset{a}{\simeq}})$ operate by congruence: they anti-unify the bodies of the shifts recursively and add the shift constructor back to the resulting pattern.

Quantifiers Rules $(\forall^{\overset{a}{\simeq}})$ and $(\exists^{\overset{a}{\simeq}})$ are symmetric. They generalize two quantified types congruently, similarly to the shift rules. However, we also require that the quantified variables are fresh, and that the left-hand side variables are equal to the corresponding variables on the right-hand side. To ensure it, alpha-renaming might be required in the implementation.

Notice that the context Θ is *not* extended with the quantified variables. In this algorithm, Θ does not play the role of a current typing context, but rather a snapshot of a context at the moment of calling the anti-unification, i.e., the context in which the instantiations of the 'holes' must be well-formed.

Functions Rule $(\rightarrow^{\overset{a}{\simeq}})$ congruently generalizes two function types. An arrow type is the only binary constructor, and thus, it is the only rule where the union of the anti-unification results is substantial. The interesting is the case when the resulting generalization of the input types and the resulting generalization of the output types have 'holes' mapped to the same pair of types. In this case, the algorithm must merge the 'holes' into one. For example, the anti-unification of $\downarrow\alpha^- \to \alpha^-$ and $\downarrow\beta^- \to \beta^-$ must result in $\downarrow\widehat{\gamma}^- \to \widehat{\gamma}^-$, rather than $\downarrow\widehat{\gamma_1}^- \to \widehat{\gamma_2}^-$.

In our representation of the anti-unification algorithm, this 'merging' happens automatically: following the rule (AU), the name of the 'hole' is uniquely defined by the pair of types it is mapped to. Specifically, when anti-unifying $\downarrow\alpha^- \to \alpha^-$ and $\downarrow\beta^- \to \beta^-$ our algorithm returns $\downarrow\widehat{\alpha}^-_{\{\alpha^-,\beta^-\}} \to \widehat{\alpha}^-_{\{\alpha^-,\beta^-\}}$, that is a renaming of $\downarrow\widehat{\gamma}^- \to \widehat{\gamma}^-$.

This way, as the output the rule returns the following tuple:

- $\widehat{\Theta}_1 \cup \widehat{\Theta}_2$—a simple union of the sets of 'holes' returned from by the recursive calls,
- $Q \to M$—the resulting pattern constructed from the patterns returned recursively.
- $\widehat{\tau}_1 \cup \widehat{\tau}'_1$ and $\widehat{\tau}_2 \cup \widehat{\tau}'_2$ — a union (in a relational sense) of the substitutions returned by the recursive calls. It is worth noting that the union is well-defined because the result of the substitution on a 'hole' is determined by the name of the 'hole'.

The Anti-Unification Rule Rule (AU) is the base case of the anti-unification algorithm. If the congruent rules are not applicable, it means that the input types have a substantially different structure, and thus, the only option is to create a 'hole'. There are three important aspects of this rule that we would like to discuss.

First, as mentioned earlier, the freshly created 'hole' has a name that is uniquely defined by the pair of input types. It is ensured by the following invariant: all the 'holes' in the algorithm have name $\widehat{\alpha}^-$ indexed by the pair of negative types it is mapped to. This way, the returning set of 'holes' is a singleton set $\{\widehat{\alpha}^-_{\{N,M\}}\}$; the resulting pattern is the 'hole' $\widehat{\alpha}^-_{\{N,M\}}$, and the mappings simply send it to the corresponding types: $\widehat{\alpha}^-_{\{N,M\}} \mapsto N$ and $\widehat{\alpha}^-_{\{N,M\}} \mapsto M$.

Second, this rule is only applicable to negative types; moreover, the input types are checked to be well-formed in the outer context Θ. This is required by the usage of anti-unification: we call it to build an existential type that would be an upper bound of two input types via abstracting some of their subexpressions under existential quantifiers. The existentials

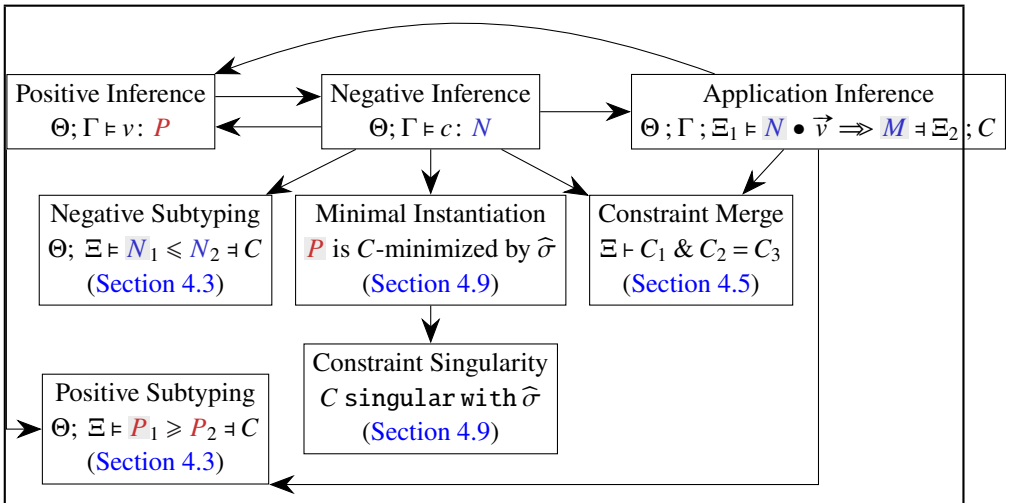

Fig. 20: Dependency graph of the typing algorithm

quantify over *negative* variables, and they must be instantiated in the context available at that moment.

Third, the rule is only applicable if all other rules fail. Notice that it could happen even when the input types have matching constructors. For example, the generalizer of $\uparrow\alpha^+$ and $\uparrow\beta^+$ is $\widehat{\gamma}^-$ (with mappings $\widehat{\gamma}^- \mapsto \uparrow\alpha^+$ and $\widehat{\gamma}^- \mapsto \uparrow\beta^+$), rather than $\uparrow\widehat{\gamma}^+$. This way, the algorithm must try to apply the congruent rules first, and only if they fail, apply (AU).

4.8 Type Inference

Finally, we present the type inference algorithm. Similarly to the subtyping algorithm, it structurally corresponds to the declarative inference specification, meaning that most of the algorithmic rules have declarative counterparts, with respect to which they are sound and complete.

This way, the inference algorithm also consists of three mutually recursive procedures: the positive type inference, the negative type inference, and the application type inference. As subroutines, the inference algorithm uses subtyping, constraint merge, and minimal instantiation. The corresponding dependency is shown in Fig. 20.

The positive and the negative type inference judgments have the symmetric forms: $\Theta; \Gamma \vDash v : P$ and $\Theta; \Gamma \vDash c : N$. Both of these algorithms take as an input typing context Θ, a variable context Γ, and a term (a value or a computation) taking its type variables from Θ, and term variables from Γ. As an output, they return a type of the given term, which we guarantee to be normalized.

The application type inference judgment has the form $\Theta ; \Gamma ; \Xi_1 \vDash N \bullet \vec{v} \Rrightarrow M \dashv \Xi_2 ; C$. As an input, it takes three contexts: typing context Θ, a variable context Γ, and an instantiation context Ξ_1. It also takes a head type N and a list of arguments (terms) $\vec{v}$ the head is applied to. The head may contain algorithmic variables specified by Ξ_1, in other words, $\Theta ; \mathrm{dom}\,(\Xi_1) \vDash N$. As a result, the application inference judgment returns M—a normalized

$\boxed{\Theta;\Gamma \vDash v\colon P}$ Positive typing

$$\frac{x\colon P \in \Gamma}{\Theta;\Gamma \vDash x\colon \mathtt{nf}\,(P)}\ (\textsc{Var}^{\textsc{inf}}) \qquad \frac{\Theta \vdash Q \quad \Theta;\Gamma \vDash v\colon P \quad \Theta;\cdot \vDash Q \geqslant P \dashv \cdot}{\Theta;\Gamma \vDash (v : Q)\colon \mathtt{nf}\,(Q)}\ (\textsc{ann}_{+}^{\textsc{inf}}) \qquad \frac{\Theta;\Gamma \vDash c\colon N}{\Theta;\Gamma \vDash \{c\}\colon \downarrow N}\ (\{\}^{\textsc{inf}})$$

$\boxed{\Theta;\Gamma \vDash c\colon N}$ Negative typing

$$\frac{\Theta \vdash M \quad \Theta;\Gamma \vDash c\colon N \quad \Theta;\cdot \vDash N \leqslant M \dashv \cdot}{\Theta;\Gamma \vDash (c : M)\colon \mathtt{nf}\,(M)}\ (\textsc{ann}_{-}^{\textsc{inf}}) \qquad \frac{\Theta;\Gamma \vDash v\colon P \quad \Theta;\Gamma, x\colon P \vDash c\colon N}{\Theta;\Gamma \vDash \mathtt{let}\, x = v;\ c\colon N}\ (\textsc{let}^{\textsc{inf}})$$

$$\frac{\Theta \vdash P \quad \Theta;\Gamma, x\colon P \vDash c\colon N}{\Theta;\Gamma \vDash \lambda x\colon P.\, c\colon \mathtt{nf}\,(P \to N)}\ (\lambda^{\textsc{inf}}) \qquad \frac{\Theta \vdash P \quad \Theta;\Gamma \vDash c\colon M \quad \Theta;\cdot \vDash M \leqslant \uparrow P \dashv \cdot \quad \Theta;\Gamma, x\colon P \vDash c'\colon N}{\Theta;\Gamma \vDash \mathtt{let}\, x\colon P = c;\ c'\colon N}\ (\textsc{let}_{\textsc{c}}^{\textsc{inf}})$$

$$\frac{\Theta, \alpha^{+};\Gamma \vDash c\colon N}{\Theta;\Gamma \vDash \Lambda\alpha^{+}.\, c\colon \mathtt{nf}\,(\forall\alpha^{+}.\, N)}\ (\Lambda^{\textsc{inf}}) \qquad \frac{\Theta;\Gamma \vDash v\colon \exists\overrightarrow{\alpha^{-}}.\, P \quad \Theta, \overrightarrow{\alpha^{-}};\Gamma, x\colon P \vDash c\colon N \quad \Theta \vdash N}{\Theta;\Gamma \vDash \mathtt{let}^{\exists}(\overrightarrow{\alpha^{-}}, x) = v;\ c\colon N}\ (\textsc{let}_{\exists}^{\textsc{inf}})$$

$$\frac{\Theta;\Gamma \vDash v\colon P}{\Theta;\Gamma \vDash \mathtt{return}\, v\colon \uparrow P}\ (\textsc{ret}^{\textsc{inf}})$$

$$\frac{\Theta \vdash P \quad \Theta;\Gamma \vDash v\colon \downarrow M \quad \Theta;\Gamma;\cdot \vDash M \bullet \vec{v} \Rrightarrow M' \dashv \Xi;C_1 \quad \Theta;\Xi \vDash M' \leqslant \uparrow P \dashv C_2 \quad \Xi \vdash C_1 \,\&\, C_2 = C \quad \Theta;\Gamma, x\colon P \vDash c\colon N}{\Theta;\Gamma \vDash \mathtt{let}\, x\colon P = v(\vec{v});\ c\colon N}\ (\textsc{let}_{:@}^{\textsc{inf}})$$

$$\frac{\Theta;\Gamma \vDash v\colon \downarrow M \quad \Theta;\Gamma;\cdot \vDash M \bullet \vec{v} \Rrightarrow \uparrow Q \dashv \Xi;C \quad Q \text{ is } C\text{-minimized by } \widehat{\sigma} \quad \Theta;\Gamma, x\colon [\widehat{\sigma}]Q \vDash c\colon N}{\Theta;\Gamma \vDash \mathtt{let}\, x = v(\vec{v});\ c\colon N}\ (\textsc{let}_{@}^{\textsc{inf}})$$

$\boxed{\Theta;\Gamma;\Xi_1 \vDash N \bullet \vec{v} \Rrightarrow M \dashv \Xi_2;C}$ Application typing

$$\frac{}{\Theta;\Gamma;\Xi \vDash N \bullet \Rrightarrow \mathtt{nf}\,(N) \dashv \Xi;\cdot}\ (\emptyset_{\bullet\Rrightarrow}^{\textsc{inf}}) \qquad \frac{\Theta;\Gamma \vDash v\colon P \quad \Theta;\Xi \vDash Q \geqslant P \dashv C_1 \quad \Theta;\Gamma;\Xi \vDash N \bullet \vec{v} \Rrightarrow M \dashv \Xi';C_2 \quad \Xi \vdash C_1 \,\&\, C_2 = C}{\Theta;\Gamma;\Xi \vDash Q \to N \bullet v, \vec{v} \Rrightarrow M \dashv \Xi';C}\ (\to_{\bullet\Rrightarrow}^{\textsc{inf}})$$

$$\frac{\Theta;\Gamma;\Xi, \overrightarrow{\widehat{\alpha^{+}}}\{\Theta\} \vDash [\overrightarrow{\widehat{\alpha^{+}}}/\overrightarrow{\alpha^{+}}]N \bullet \vec{v} \Rrightarrow M \dashv \Xi';C \quad \overrightarrow{\widehat{\alpha^{+}}} \text{ are fresh} \quad \vec{v} \neq \quad \overrightarrow{\alpha^{+}} \neq \cdot}{\Theta;\Gamma;\Xi \vDash \forall\overrightarrow{\alpha^{+}}.\, N \bullet \vec{v} \Rrightarrow M \dashv \Xi';C|_{\mathtt{fav}(N)\cup\mathtt{fav}(M)}}\ (\forall_{\bullet\Rrightarrow}^{\textsc{inf}})$$

Fig. 21: Algorithmic Type Inference

type of the result of the application. Type M may contain new algorithmic variables, and thus, the judgment also returns Ξ_2—an updated instantiation context and C—a set of subtyping constraints. Together Ξ_2 and C specify how the algorithmic variables must be instantiated.

The inference rules are shown in Fig. 21. Next, we discuss them in detail.

Variables Rule ($\textsc{Var}^{\textsc{inf}}$) infers the type of a positive variable by looking it up in the term variable context and normalizing the result.

Annotations Rules ($\textsc{ann}_{+}^{\textsc{inf}}$) and ($\textsc{ann}_{-}^{\textsc{inf}}$) are symmetric. First, they check that the annotated type is well-formed in the given context Θ. Then they make a recursive call to infer the type of annotated expression, check that the inferred type is a subtype of the annotation, and return the normalized annotation.

Abstractions Rule ($\lambda^{\textsc{inf}}$) infers the type of a lambda abstraction. It checks the well-formedness of the annotation P, makes a recursive call to infer the type of the body in the extended context, and returns the corresponding arrow type. Since the annotation P is allowed to be non-normalized, the rule also normalizes the resulting type.

Rule ($\Lambda^{\textsc{inf}}$) infers the type of a big lambda. Similarly to the previous case, it makes a recursive call to infer the type of the body in the extended *type* context. After that, it returns the corresponding universal type. It is also required to normalize the result. For instance, if α^{+} does not occur in the body of the lambda, the corresponding $\forall$ will be removed.

Return and Thunk Rules ($\{\}^{\textsc{inf}}$) and ($\textsc{ret}^{\textsc{inf}}$) are similar to the declarative rules: they make a recursive call to type the body of the thunk or the return expression and put the shift on top of the result.

Unpack Rule ($\textsc{let}_{\exists}^{\textsc{inf}}$) allows one to unpack an existential type. First, it infers the existential type $\exists\overrightarrow{\alpha^{-}}.\ P$ of the value being unpacked, and since the type is guaranteed to be normalized, binds the quantified variables with $\overrightarrow{\alpha^{-}}$. Then it infers the type of the body in the appropriately extended context and checks that the inferred type does not depend on $\overrightarrow{\alpha^{-}}$ by checking well-formedness $\Theta \vdash N$.

Let Binders Rule ($\textsc{let}^{\textsc{inf}}$) represents the type inference of a standard let binder. It infers the type of the bound value v, and makes a recursive call to infer the type of the body in the extended context.

Rule ($\textsc{let}_{\mathrm{c}}^{\textsc{inf}}$) infers a type of computational let binder. It follows the corresponding declarative rule ($\textsc{let}_{\mathrm{c}}^{\textsc{inf}}$) but uses algorithmic judgments instead of declarative ones. It is worth noting that when calling the subtyping $\Theta;\ \cdot \vDash M \leqslant \uparrow P \dashv \cdot$, both M and P are free of algorithmic variables: M is a type inferred for c, and P is given as an annotation.

Rule ($\textsc{let}_{:@}^{\textsc{inf}}$) infers a type of *annotated* applicative let binder. First, it infers the type of the head of the application, ensuring that it is a *thunked computation* $\downarrow M$. After that, it makes a recursive call to the application inference procedure, returning an algorithmic type M', that must be a subtype of the annotation $\uparrow P$.

Then premise $\Theta;\ \Xi \vDash M' \leqslant \uparrow P \dashv C_2$ together with $\Xi \vdash C_1 \,\&\, C_2 = C$ check that M' can be instantiated to the annotated type $\uparrow P$, and if it is, the algorithm infers the type of the body in the extended context, and returns it as the result.

Rule ($\textsc{let}_{@}^{\textsc{inf}}$) works similarly to ($\textsc{let}_{:@}^{\textsc{inf}}$). However, since no annotation is given, the algorithm must ensure that the inferred Q has the 'canonical' minimal instantiation. To find it, it makes a call to the minimal instantiation algorithm (Section 4.9) that finds the substitution that satisfies the inferred constraints C and instantiates Q to the minimal (among other such instantiations) type $[\widehat{\sigma}]\,Q$.

Application to an Empty List of Arguments Rule ($\emptyset^{\text{INF}}_{\bullet\Rightarrow}$) is the base case of application inference. If the list of arguments is empty, the inferred type is the type of the head, and the algorithm returns it after normalizing.

Application of a Polymorphic Type $\forall$ Rule ($\forall^{\text{INF}}_{\bullet\Rightarrow}$), analogously to the declarative case, is the rule ensuring the implicit elimination of the universal quantifiers. This is the place where the algorithmic variables are introduced. The algorithm simply replaces the quantified variables $\overrightarrow{\alpha^+}$ with fresh algorithmic variables $\overrightarrow{\widehat{\alpha}^+}$, and makes a recursive call in the extended context.

To ensure the rule precedence, we also require the head type to have at least one $\forall$-quantifier, and the list of arguments to be non-empty.

Application of an Arrow Type Rule ($\to^{\text{INF}}_{\bullet\Rightarrow}$) is the main rule of algorithmic application inference. It is applied when the head has an arrow type $Q \to N$. First, it infers the type of the first argument v, and then, calling the algorithmic subtyping, finds C_1—the *minimal* constraint ensuring that Q is a supertype of the type of v. Then it makes a recursive call applying N to the rest of the arguments and merges the resulting constraint with C_1.

4.9 Minimal Instantiation and Constraint Singularity

Multiple types M can be inferred for a type application: $\Theta\,;\Gamma \vdash N \bullet \vec{v} \Rrightarrow M$ but only one *principal* type should be chosen for a variable in an unannotated let binder. Declaratively, we require the principal type P to be minimal among other all types P' that can be inferred for the application $\Theta\,;\Gamma \vdash N \bullet \vec{v} \Rrightarrow \uparrow P'$. Algorithmically, the inference returns an *algorithmic* type P together with a set of *necessary and sufficient* constraints C that any instantiation of P must satisfy. This way, the principal type is the minimal instantiation of P, obtained by a substitution satisfying the given constraints C.

To find this substitution, we use the minimal instantiation algorithm (Fig. 22). In the judgment "P is C-minimized by $\widehat{\sigma}$", $\widehat{\sigma}$ instantiates P to a subtype of any other instantiation of P that satisfies C. First, it removes the existential quantifiers by ($\exists^{\text{MIN}}$). Then it considers two cases.

1. If the type P is an algorithmic variable $\widehat{\alpha}^+$ restricted by a *subtyping* constraint $(\widehat{\alpha}^+ :\geqslant Q) \in C$, its minimal instantiation is the type $\mathtt{nf}\,(Q)$, and (UVar^{MIN}) is applied.

2. Otherwise, the type P is either a shifted computation, a declarative variable, an unrestricted algorithmic variable, or an algorithmic variable restricted by an *equivalence* constraint. In all of these cases, the minimal instantiation exists if and only if there is only one possible instantiation of P that satisfies C.

   The algorithm finds this instantiation by two premises: (i) checking that all the algorithmic variables of the type are restricted by the constraint set; and (ii) building the substitution that satisfies the constraint set on these variables (and simultaneously checking that such substitution is unique up to equivalence) using the *constraint singularity algorithm*.

The singularity algorithm performs two tasks: it checks that a *constraint set* has a single substitution satisfying it, and if it does, it builds this substitution.

$$\frac{(\widehat{\alpha}^{+} :\geqslant P) \in C}{\widehat{\alpha}^{+} \text{ is } C\text{-minimized by } (\mathtt{nf}\,(P)/\widehat{\alpha}^{+})}\ (\text{UVAR}^{\text{MIN}}) \qquad \frac{\mathtt{fav}(P) \subseteq \mathtt{dom}\,(C) \quad C|_{\mathtt{fav}(P)}\ \mathtt{singular\ with}\ \widehat{\sigma}}{P \text{ is } C\text{-minimized by } \widehat{\sigma}}\ (\text{SING}^{\text{MIN}})$$

$$\frac{P \text{ is } C\text{-minimized by } \widehat{\sigma}}{\exists\overrightarrow{\alpha^{-}}.\ P \text{ is } C\text{-minimized by } \widehat{\sigma}}\ (\exists^{\text{MIN}})$$

Fig. 22: Minimal Instantiation

'C $\mathtt{singular\ with}$ $\widehat{\sigma}$' means:

+ for any *positive* constraint entry $e \in C$ restricting a variable $\widehat{\beta}^{+}$, there exists P such that e $\mathtt{singular\ with}$ P (as defined in Fig. 24), and $[\widehat{\sigma}]\widehat{\beta}^{+} = P$; and
− the symmetric property holds for all *negative* $e \in C$; and
∉ for any $\widehat{\alpha}^{\pm} \notin \mathtt{dom}\,(C)$, $[\widehat{\sigma}]\widehat{\alpha}^{\pm} = \widehat{\alpha}^{\pm}$.

Fig. 23: Singular Constraint

$$\frac{}{\widehat{\alpha}^{+} :\simeq P\ \mathtt{singular\ with\ nf}\,(P)}\ (\simeq^{\text{SING}}_{+}) \qquad \frac{}{\widehat{\alpha}^{-} :\simeq N\ \mathtt{singular\ with\ nf}\,(N)}\ (\simeq^{\text{SING}}_{-})$$

$$\frac{}{\widehat{\alpha}^{+} :\geqslant \exists\overrightarrow{\alpha^{-}}.\ \alpha^{+}\ \mathtt{singular\ with}\ \alpha^{+}}\ (:\geqslant\alpha^{\text{SING}}) \qquad \frac{\mathtt{nf}\,(N) = \alpha_i^{-} \in \overrightarrow{\alpha^{-}}}{\widehat{\alpha}^{+} :\geqslant \exists\overrightarrow{\alpha^{-}}.\ \downarrow N\ \mathtt{singular\ with}\ \exists\beta^{-}.\ \downarrow\beta^{-}}\ (:\geqslant\downarrow^{\text{SING}})$$

Fig. 24: Singular Constraint Entry

To implement the singularity algorithm, we define a partial function 'C $\mathtt{singular\ with}$ $\widehat{\sigma}$', taking a subtyping constraint C as an argument and returning a substitution $\widehat{\sigma}$—the only solution of C.

The constraint C is composed of constraint entries. Therefore, we define singularity by combining the singularity of each constraint entry (Fig. 23). In order for C to be singular, each entry must have a unique instantiation, and the resulting substitution $\widehat{\sigma}$ must be a union of these instantiations. In addition, $\widehat{\sigma}$ must act as an identity on the variables not restricted by C.

The singularity of constraint *entries* is defined in Fig. 24.

- The *equivalence* entries are always singular, as the only possible type satisfying them is the one given in the constraint itself, which is reflected in rules $(\simeq^{\text{SING}}_{-})$ and $(\simeq^{\text{SING}}_{+})$.
- The *subtyping* constraints are trickier. As will be discussed in Section 5.3, variables (and equivalent them $\exists\overrightarrow{\alpha^{-}}.\ \alpha^{+}$) do not have proper supertypes, and thus, the constraints of a form $\widehat{\alpha}^{+} :\geqslant \exists\overrightarrow{\alpha^{-}}.\ \alpha^{+}$ are singular with the only possible normalized solution α^{+}—see rule $(:\geqslant\alpha^{\text{SING}})$.
- However, if the body of the existential type is guarded by a shift $\exists\overrightarrow{\alpha^{-}}.\ \downarrow N$, it is singular if and only if N is equivalent to some $\alpha_i^{-} \in \overrightarrow{\alpha^{-}}$ bound by the quantifier—see rule $(:\geqslant\downarrow^{\text{SING}})$. The completeness of this criterion is justified by the fact that if N is

not equivalent to any α_i^-, then there are two non-equivalent solutions of constraint $(\widehat{\alpha}^+ :\geqslant \exists\overrightarrow{\alpha^-}.\downarrow N)$: the trivial $\exists\overrightarrow{\alpha^-}.\downarrow N$ and $\exists\overrightarrow{\alpha^-}.\downarrow\alpha_1^-$ (which is a supertype of $\exists\overrightarrow{\alpha^-}.\downarrow N$ since α_1^- can be instantiated to N).

5 Algorithm Correctness

The central results ensuring the correctness of the inference algorithm are its soundness and completeness with respect to the declarative specification. The soundness means the algorithm will always produce a typing *allowed* by the declarative system; dually, the completeness says that once a term has some type declaratively, the inference algorithm succeeds.

The formal statements of soundness and completeness are given in the theorems below. Notice that the theorems also include the soundness and completeness of *application inference* (labeled as •), which is more complex. As such, let us discuss it in more detail.

Both soundness and completeness of application inference assume that the input head type N is free from *negative* algorithmic variables—it is achieved by polarization invariants preserved by the inference rules. The soundness states that the output of the algorithm—M and C—is viable. Specifically, that the constraint set C provides a sufficient set of restrictions that any substitution $\widehat{\sigma}$ must satisfy to ensure the *declarative* inference of the output type M, that is $\Theta\,;\Gamma \vdash [\widehat{\sigma}]N \bullet \vec{v} \Longrightarrow [\widehat{\sigma}]M$.

The application inference completeness means that if there exists a substitution $\widehat{\sigma}$ and the resulting type M ensuring the declarative inference $\Theta\,;\Gamma \vdash [\widehat{\sigma}]N \bullet \vec{v} \Longrightarrow M$ then the algorithm succeeds and gives the most general result M_0 and C_0. The property of 'being the most general' is specified in pt. (2). Intuitively, it means that for any other solution—substitution $\widehat{\sigma}$ and the resulting type M, if it ensures the declarative inference, then $\widehat{\sigma}$ can be extended in a C_0-complying way to equate M_0 with M.

Theorem (Soundness of Typing)**.** *Suppose that* $\Theta \vdash \Gamma$. *Then*[1]

- \+ $\Theta;\Gamma \vDash v\colon P$ *implies* $\Theta;\Gamma \vdash v\colon P$,
- – $\Theta;\Gamma \vDash c\colon N$ *implies* $\Theta;\Gamma \vdash c\colon N$,
- • $\Theta\,;\Gamma\,;\Xi \vDash N \bullet \vec{v} \Longrightarrow M \dashv \Xi'\,;C$ *implies* $\Theta\,;\Gamma \vdash [\widehat{\sigma}]N \bullet \vec{v} \Longrightarrow [\widehat{\sigma}]M$, *for any instantiation of* $\widehat{\sigma}$ *satisfying constraints* C. *All of it under assumptions that* $\Theta \vdash^{\supseteq} \Xi$ *and* $\Theta\,;\mathtt{dom}\,(\Xi) \vdash N$ *and that* N *is free from* negative *algorithmic variables.*

Theorem (Completeness of Typing)**.** *Suppose that* $\Theta \vdash \Gamma$. *Then*[1]

- \+ $\Theta;\Gamma \vdash v\colon P$ *implies* $\Theta;\Gamma \vDash v\colon \mathtt{nf}\,(P)$,
- – $\Theta;\Gamma \vdash c\colon N$ *implies* $\Theta;\Gamma \vDash c\colon \mathtt{nf}\,(N)$,
- • *If* $\Theta\,;\Gamma \vdash [\widehat{\sigma}]N \bullet \vec{v} \Longrightarrow M$ *where 1.* $\Theta \vdash^{\supseteq} \Xi$, *2.* $\Theta \vdash M$, *3.* $\Theta\,;\mathtt{dom}\,(\Xi) \vdash N$ *(free from* negative *algorithmic variables), and 4.* $\Xi \vdash \widehat{\sigma} : \mathtt{fav}(N)$, *then there exist* M_0, Ξ_0, *and* C_0 *such that*

[1] The presented properties hold, but the actual inductive proof requires strengthening of the statement and the corresponding theorem is more involved. See the appendix (Section B.4.15) for details.

1. $\Theta ; \Gamma ; \Xi \vDash N \bullet \vec{\nu} \Rrightarrow M_0 \dashv \Xi_0 ; C_0$ *and*

2. *for any other* $\widehat{\sigma}$ *and* M *(where* $\Xi \vdash \widehat{\sigma} : \mathtt{fav}(N)$ *and* $\Theta \vdash M$*) such that* $\Theta ; \Gamma \vdash [\widehat{\sigma}]N \bullet \vec{\nu} \Rrightarrow M$*, there exists* $\widehat{\sigma}'$ *such that a.* $\Xi_0 \vdash \widehat{\sigma}' : \mathtt{fav} N \cup \mathtt{fav} M_0$ *and* $\Xi_0 \vdash \widehat{\sigma}' : C_0$*, b.* $\Xi \vdash \widehat{\sigma}' \simeq^{\leqslant} \widehat{\sigma} : \mathtt{fav} N$*, and c.* $\Theta \vdash [\widehat{\sigma}']M_0 \simeq^{\leqslant} M$.

The proof of soundness and completeness result is done gradually for all the subroutines, following the structure of the algorithm (Figs. 13 and 20) bottom-up. Next, we discuss the main results.

5.1 Normalization

The point of type normalization is factoring out non-trivial equivalence by selecting a representative from each equivalence class. This way, *the correctness of normalization* means that checking for the equivalence of two types is the same as checking for the equality of their normal forms.

Lemma (Normalization correctness)**.** *Assuming all types are well-formed in* Θ*, we have* $\Theta \vdash N \simeq^{\leqslant} M \iff \mathtt{nf}(N) = \mathtt{nf}(M)$ *and* $\Theta \vdash P \simeq^{\leqslant} Q \iff \mathtt{nf}(P) = \mathtt{nf}(Q)$.

To prove the correctness of normalization, we introduce an *intermediate* relation on types—*declarative equivalence* (the notation is $N \simeq^{D} M$ and $P \simeq^{D} Q$). In contrast to $\Theta \vdash N \simeq^{\leqslant} M$ (which means mutual subtyping), $N \simeq^{D} M$ does not depend on subtyping judgments, but explicitly allows quantifier reordering and redundant quantifier removal. Then the statement $\Theta \vdash N \simeq^{\leqslant} M \iff \mathtt{nf}(N) = \mathtt{nf}(M)$ (as well as its positive counterpart) is split into two steps: $\Theta \vdash N \simeq^{\leqslant} M \iff N \simeq^{D} M \iff \mathtt{nf}(N) = \mathtt{nf}(M)$.

5.2 Anti-Unification

The anti-unifier of the two types is the most specific pattern that matches against both of them. In our setting, the anti-unifiers are restricted further: first, the pattern might only contain 'holes' at *negative* positions (because eventually, the 'holes' become variables abstracted by the existential quantifier); second, the anti-unification is parametrized with a context Θ, in which the pattern instantiations must be well-formed.

This way, the correctness properties of the anti-unification algorithm are refined accordingly. The soundness of anti-unification not only ensures that the resulting pattern matches with the input types, but also that the pattern instantiations are well-formed in the corresponding context, and that all the 'hole' variables are negative. In turn, completeness states that if there exists a solution satisfying the soundness criteria, then the algorithm succeeds.

The correctness properties are formulated by the following lemmas. For brevity, we only provide the statements for the positive case, since the negative case is symmetric.

Lemma (Soundness of (positive) anti-unification)**.** *Assuming* P_1 *and* P_2 *are normalized, if* $\Theta \vDash P_1 \overset{a}{\simeq} P_2 \dashv (\widehat{\Theta}, Q, \widehat{\tau}_1, \widehat{\tau}_2)$ *then 1.* $\Theta ; \widehat{\Theta} \vdash Q$*, 2.* $\Theta ; \cdot \vdash \widehat{\tau}_i : \widehat{\Theta}$ *for* $i \in \{1, 2\}$ *are anti-unification substitutions (in particular,* $\widehat{\Theta}$ *contains only negative algorithmic variables), and 3.* $[\widehat{\tau}_i]Q = P_i$ *for* $i \in \{1, 2\}$.

Lemma (Completeness of (positive) anti-unification)**.** *Assuming that* P_1 *and* P_2 *are normalized terms well-formed in* Θ *and there exist* $(\widehat{\Theta}', Q', \widehat{\tau}_1', \widehat{\tau}_2')$ *such that 1.* $\Theta ; \widehat{\Theta}' \vdash Q'$*, 2.* $\Theta ; \cdot \vdash \widehat{\tau}_i' : \widehat{\Theta}'$ *for* $i \in \{1, 2\}$ *are anti-unification substitutions, and 3.* $[\widehat{\tau}_i'] Q' = P_i$ *for* $i \in \{1, 2\}$*.*

Then the anti-unification algorithm terminates, that is there exists $(\widehat{\Theta}, Q, \widehat{\tau}_1, \widehat{\tau}_2)$ *such that* $\Theta \vDash P_1 \overset{a}{\simeq} P_2 \dashv (\widehat{\Theta}, Q, \widehat{\tau}_1, \widehat{\tau}_2)$*.*

Notice that for the anti-unification substitution $\widehat{\tau}$ we use a separate signature specifying the domain and the range. $\Theta ; \widehat{\Theta}_2 \vdash \widehat{\tau} : \widehat{\Theta}_1$ means that $\widehat{\tau}$ maps the 'holes' (i.e., algorithmic variables) from $\widehat{\Theta}_1$ to *algorithmic* types well-formed in Θ and $\widehat{\Theta}_2$. To put it formally, $\Theta ; \widehat{\Theta}_2 \vdash [\widehat{\tau}]\widehat{\alpha}^-$ for any $\widehat{\alpha}^- \in \widehat{\Theta}_1$. Although in the formulation of soundness and completeness, the resulting types are declarative (i.e., $\widehat{\Theta}_2$ is always empty), we need the possibility of mapping the 'holes' to types with 'holes' to formulate the *initiality* of anti-unification.

The initiality shows that the anti-unifier that the algorithm provides is the most specific (or the most 'detailed'). Specifically, it means that any other sound anti-unification solution can be 'refined' to the result of the algorithm. The 'refinement' is represented as an instantiation of the anti-unifier—a substitution $\Theta ; \widehat{\Theta}_2 \vdash \widehat{\rho} : \widehat{\Theta}_1$—replacing the 'holes' from $\widehat{\Theta}_1$ with types that themselves can contain 'holes' from $\widehat{\Theta}_2$.

Lemma (Initiality of Anti-Unification)**.** *Assume that* P_1 *and* P_2 *are normalized types well-formed in* Θ*, and the anti-unification algorithm succeeds:* $\Theta \vDash P_1 \overset{a}{\simeq} P_2 \dashv (\widehat{\Theta}, Q, \widehat{\tau}_1, \widehat{\tau}_2)$*. Then* $(\widehat{\Theta}, Q, \widehat{\tau}_1, \widehat{\tau}_2)$ *is more specific than any other sound anti-unifier* $(\widehat{\Theta}', Q', \widehat{\tau}_1', \widehat{\tau}_2')$*, i.e., if 1.* $\Theta ; \widehat{\Theta}' \vdash Q'$*, 2.* $\Theta ; \cdot \vdash \widehat{\tau}_i' : \widehat{\Theta}'$ *for* $i \in \{1, 2\}$*, and 3.* $[\widehat{\tau}_i'] Q' = P_i$ *for* $i \in \{1, 2\}$ *then there exists a 'refining' substitution* $\widehat{\rho}$ *such that* $\Theta ; \widehat{\Theta} \vdash \widehat{\rho} : (\widehat{\Theta}'|_{\mathtt{fav}\, Q'})$ *and* $[\widehat{\rho}] Q' = Q$*.*

To prove the correctness properties of the anti-unification algorithm, one extra observation is essential. The algorithm relies on the fact that in the resulting tuple $(\widehat{\Theta}, Q, \widehat{\tau}_1, \widehat{\tau}_2)$, there are no two different 'holes' $\widehat{\beta}^-$ mapped to the same pair of types by $\widehat{\tau}_1$ and $\widehat{\tau}_2$. This is used to ensure that, for example, the anti-unifier of $\downarrow\uparrow\mathtt{Int} \to \uparrow\mathtt{Int}$ and $\downarrow\uparrow\mathtt{Bool} \to \uparrow\mathtt{Bool}$ is $\downarrow\widehat{\alpha}^- \to \widehat{\alpha}^-$, but not (less specific) $\downarrow\widehat{\alpha}^- \to \widehat{\beta}^-$. To preserve this property, we arrange the algorithm in such a way that the name of the 'hole' is determined by the types it is mapped to. The following lemma specifies this observation.

Lemma (Uniqueness of Anti-unification Variable Names)**.** *Names of the anti-unification variables are uniquely defined by the types they are mapped to by the resulting substitutions. Assuming* P_1 *and* P_2 *are normalized, if* $\Theta \vDash P_1 \overset{a}{\simeq} P_2 \dashv (\widehat{\Theta}, Q, \widehat{\tau}_1, \widehat{\tau}_2)$ *then for any* $\widehat{\beta}^- \in \widehat{\Theta}$*,* $\widehat{\beta}^- = \widehat{\alpha}^-_{\{[\widehat{\tau}_1]\widehat{\beta}^-, [\widehat{\tau}_2]\widehat{\beta}^-\}}$*.*

5.3 Least Upper Bound and Upgrade

The Least Upper Bound algorithm finds the least type that is a supertype of two given types. Its *soundness* means that the returned type is indeed a supertype of the given ones; the *completeness* means that the algorithm succeeds if the least upper bound exists; and the *initiality* means that the returned type is the least among the common supertypes.

Lemma (Least Upper Bound soundness)**.** *For types* $\Theta \vdash P_1$*, and* $\Theta \vdash P_2$*, if* $\Theta \vDash P_1 \vee P_2 = Q$ *then*

(i) $\Theta \vdash Q$

(ii) $\Theta \vdash Q \geqslant P_1$ *and* $\Theta \vdash Q \geqslant P_2$

Lemma (Least Upper Bound completeness and initiality)**.** *For types* $\Theta \vdash P_1$, $\Theta \vdash P_2$, *and* $\Theta \vdash Q$ *such that* $\Theta \vdash Q \geqslant P_1$ *and* $\Theta \vdash Q \geqslant P_2$, *there exists* Q' *s.t.* $\Theta \vDash P_1 \vee P_2 = Q'$ *and* $\Theta \vdash Q \geqslant Q'$.

The key observation that allows us to prove these properties is the characterization of positive supertypes. The following lemma justifies the case analysis used in the Least Upper Bound algorithm (in Section 4.6). In particular, it establishes the correspondence between the upper bounds of shifted types $\downarrow M$ and *patterns* fitting M (represented by existential types), which explains the usage of anti-unification as a way to find a common pattern.

Lemma (Characterization of normalized supertypes)**.** *For a normalized positive type* P *well-formed in* Θ, *the set of normalized* Θ*-formed supertypes of* P *is the following:*

- *if* P *is a variable* β^+, *its only normalized supertype is* β^+ *itself;*
- *if* P *is an existential type* $\exists \overrightarrow{\beta^-}.\, P'$ *then its* Θ*-formed supertypes are the* $(\Theta, \overrightarrow{\beta^-})$*-formed supertypes of* P' *not using* $\overrightarrow{\beta^-}$*;*
- *if* P *is a downshift type* $\downarrow M$, *its supertypes have form* $\exists \overrightarrow{\alpha^-}.\, \downarrow M'$ *such that there exists a* Θ*-formed instantiation of* $\overrightarrow{\alpha^-}$ *in* $\downarrow M'$ *that makes* $\downarrow M'$ *equal to* $\downarrow M$, *i.e.,* $[\overrightarrow{N}/\overrightarrow{\alpha^-}]\downarrow M' = \downarrow M$.

Similarly to the Least Upper Bound algorithm, the Upgrade finds the least type among upper bounds (this time the set of considered upper bounds consists of supertypes well-formed in a *smaller* context). This way, we also use the supertype characterization to prove the following properties of the Upgrade algorithm.

Lemma (Upgrade soundness)**.** *Assuming* P *is well-formed in* $\Theta = \Theta_0, \overrightarrow{\alpha^{\pm}}$, *if* $\texttt{upgrade}\ \Theta \vdash P\ \texttt{to}\ \Theta_0 = Q$ *then 1.* $\Theta_0 \vdash Q$ *and 2.* $\Theta \vdash Q \geqslant P$.

Lemma (Upgrade Completeness)**.** *Assuming* P *is well-formed in* $\Theta = \Theta_0, \overrightarrow{\alpha^{\pm}}$, *for any* Q' *such that* Q' *is a* Θ_0*-formed upper bound of* P, *i.e., 1.* $\Theta_0 \vdash Q'$ *and 2.* $\Theta \vdash Q' \geqslant P$, *there exists* Q *such that (*$\texttt{upgrade}\ \Theta \vdash P\ \texttt{to}\ \Theta_0 = Q$*) and* $\Theta_0 \vdash Q' \geqslant Q$.

5.4 Subtyping

As for other properties, the correctness of subtyping means that the algorithm produces a valid result (soundness) whenever it exists (completeness). In the rules defining the subtyping algorithm (Fig. 14), one can see that the positive and negative subtyping relations are not *mututally* recursive: negative subtyping algorithm uses the positive subtyping, but not vice versa. Because of that, the inductive proofs of the *positive* subtyping correctness are done independently.

The soundness of positive subtyping states that the output constraint C provides a sufficient set of restrictions. In other words, any substitution satisfying C ensures the desired declarative subtyping: $\Theta \vdash [\widehat{\sigma}]\, P \geqslant Q$. Notice that the soundness formulation assumes that only the

left-hand side input type (P) can contain algorithmic variables. This is one of the invariants of the algorithm that significantly simplifies the unification and constraint resolution.

Lemma (Soundness of Positive Subtyping). *If* $\Theta \vdash^{\supseteq} \Xi$ *and* $\Theta \vdash Q$ *and* $\Theta\,;\mathtt{dom}\,(\Xi) \vdash P$ *and* $\Theta;\ \Xi \vDash P \geqslant Q \dashv C$, *then* $\Xi \vdash C : \mathtt{fav} P$ *and for any* $\widehat{\sigma}$ *such that* $\Xi \vdash \widehat{\sigma} : C$, *we have* $\Theta \vdash [\widehat{\sigma}]P \geqslant Q$.

The completeness of subtyping says that if the substitution ensuring the declarative subtyping exists, then the subtyping algorithm succeeds (terminates).

Lemma (Completeness of Positive Subtyping). *Suppose that* $\Theta \vdash^{\supseteq} \Xi$, $\Theta \vdash Q$ *and* $\Theta\,;\mathtt{dom}\,(\Xi) \vdash P$. *Then if there exists* $\widehat{\sigma}$ *such that* $\Xi \vdash \widehat{\sigma} : \mathtt{fav}(P)$ *and* $\Theta \vdash [\widehat{\sigma}]P \geqslant Q$, *then the subtyping algorithm succeeds:* $\Theta;\ \Xi \vDash P \geqslant Q \dashv C$.

After the correctness properties of positive subtyping are established, they are used to prove the correctness of the negative subtyping. The soundness is formulated symmetrical to the positive case, however, the completeness requires an additional invariant to be preserved. The algorithmic input type N must be free from *negative* algorithmic variables. In particular, it ensures that the constraint restricting a *negative* algorithmic variable will never be generated, and thus, we do not need the Greatest Common Subtype procedure to resolve the constraints.

Lemma (Completeness of Negative Subtyping). *Suppose that* $\Theta \vdash^{\supseteq} \Xi$ *and* $\Theta \vdash M$ *and* $\Theta\,;\mathtt{dom}\,(\Xi) \vdash N$ *and* N *does not contain negative unification variables* ($\widehat{\alpha}^- \notin \mathtt{fav} N$). *Then for any* $\Xi \vdash \widehat{\sigma} : \mathtt{fav}(N)$ *such that* $\Theta \vdash [\widehat{\sigma}]N \leqslant M$, *the subtyping algorithm succeeds:* $\Theta;\ \Xi \vDash N \leqslant M \dashv C$.

5.5 Minimal Instantiation and Singularity

Algorithmic typing relies on the *minimal instantiation* procedure. For a given positive type P and a set of constraints C it returns a substitution $\Xi \vdash \widehat{\sigma} : C$ such that $[\widehat{\sigma}]P$ is a subtype of all the other instantiations of P satisfying C. The correctness of minimal instantiation is formulated as the following two lemmas.

Lemma (Soundness of Minimal Instantiation). *Suppose that* $\Theta \vdash^{\supseteq} \Xi$, $\Xi \vdash C$, *and* $\Theta\,;\mathtt{dom}\,(\Xi) \vdash P$. *Then the judgment '*$P$ *is* C*-minimized by* $\widehat{\sigma}$*' implies that* $\Xi \vdash \widehat{\sigma} : \mathtt{fav} P$ *is a normalized substitution satisfying* C *(i.e.,* $\Xi \vdash \widehat{\sigma} : C$*) and for any other substitution* $\Xi \vdash \widehat{\sigma}' : \mathtt{fav} P$ *satisfying* C, *we have* $\Theta \vdash [\widehat{\sigma}']P \geqslant [\widehat{\sigma}]P$.

Lemma (Completeness of Minimal Instantiation). *Suppose that* $\Theta \vdash^{\supseteq} \Xi$, $\Xi \vdash C$, $\Theta\,;\mathtt{dom}\,(\Xi) \vdash P$, *and there exists* $\Xi \vdash \widehat{\sigma} : \mathtt{fav} P$ *satisfying* C ($\Xi \vdash \widehat{\sigma} : C$) *such that for any other* $\Xi \vdash \widehat{\sigma}' : \mathtt{fav} P$ *satisfying* C, *we have* $\Theta \vdash [\widehat{\sigma}']P \geqslant [\widehat{\sigma}]P$. *Then the minimal instantiation procedure succeeds; specifically,* P *is* C*-minimized by* $\mathtt{nf}\,(\widehat{\sigma})$ *is derivable.*

Soundness is rather straightforwardly proved by induction on the judgment inference tree, relying on the soundness of the singularity procedure. The proof of completeness is done by induction on the structure of the type P. It uses the fact that $\Theta \vdash \exists\overrightarrow{\alpha^-}.\ [\widehat{\sigma}']P \geqslant \exists\overrightarrow{\alpha^-}.\ [\widehat{\sigma}]P$ implies $\Theta, \overrightarrow{\alpha^-} \vdash [\widehat{\sigma}']P \geqslant [\widehat{\sigma}]P$.

The soundness of *singularity* states that C $\mathtt{singular\ with}$ $\widehat{\sigma}$ implies that any substitution satisfying C is equivalent to $\widehat{\sigma}$ on the domain. The completeness states that if all the C-compliant substitutions are equivalent, then the singularity procedure succeeds.

Lemma (Soundness of Singularity). *Suppose* $\Xi \vdash C : \widehat{\Theta}$, *and* C `singular with` $\widehat{\sigma}$. *Then* $\Xi \vdash \widehat{\sigma} : \widehat{\Theta}$, $\Xi \vdash \widehat{\sigma} : C$ *and for any* $\widehat{\sigma}'$ *such that* $\Xi \vdash \widehat{\sigma} : C$, $\Xi \vdash \widehat{\sigma}' \simeq^{\leqslant} \widehat{\sigma} : \widehat{\Theta}$.

Lemma (Completeness of Singularity). *For a given set of constraints* $\Xi \vdash C$, *suppose that all the substitutions satisfying* C *are equivalent on* $\mathtt{dom}\,(C)$. *In other words, suppose that there exists* $\Xi \vdash \widehat{\sigma}_1 : \mathtt{dom}\,(C)$ *such that for any* $\Xi \vdash \widehat{\sigma} : \mathtt{dom}\,(C)$, $\Xi \vdash \widehat{\sigma} : C$ *implies* $\Xi \vdash \widehat{\sigma} \simeq^{\leqslant} \widehat{\sigma}_1 : \mathtt{dom}\,(C)$. *Then* C `singular with` $\widehat{\sigma}_0$ *for some* $\widehat{\sigma}_0$.

5.6 Typing

Finally, we discuss the proofs of the soundness and completeness of the type inference algorithm that we described at the beginning of this section. There are three subtleties that are important for the proof to go through: the determinacy of the algorithm, the mutual dependence of the soundness and completeness proofs, and the non-trivial inductive metric that we use.

Determinacy One of the properties that our proof relies on is the determinacy of the typing algorithm: the output (the inferred type) is uniquely determined by the input (the term and the contexts). Determinacy is not hard to demonstrate by structural induction: in every algorithmic inference system, only one inference rule can be applied for a given input. However, we need to prove it for *every* subroutine of the algorithm. Ultimately, it requires the determinacy of such procedures as *generation of fresh variables*, which is easy to ensure, but must be taken into account in the implementation.

Lemma (Determinacy of the Typing Algorithm). *Suppose that* $\Theta \vdash \Gamma$ *and* $\Theta \vdash^{\supseteq} \Xi_0$. *Then*

- *If* $\Theta; \Gamma \vDash v : P$ *and* $\Theta; \Gamma \vDash v : P'$ *then* $P = P'$.
- *If* $\Theta; \Gamma \vDash c : N$ *and* $\Theta; \Gamma \vDash c : N'$ *then* $N = N'$.
- *If* $\Theta ; \Gamma ; \Xi_0 \vDash N \bullet \vec{v} \Rightarrow\!\!> M \dashv \Xi ; C$ *and* $\Theta ; \Gamma ; \Xi_0 \vDash N \bullet \vec{v} \Rightarrow\!\!> M' \dashv \Xi' ; C'$ *then* $M = M'$, $\Xi = \Xi'$, *and* $C = C'$.

Mutuality of the Soundness and Completeness Proofs Typically in our inductive proofs, the soundness is proven before completeness, as the completeness requires the premise subtrees to satisfy certain properties (which can be given by the soundness). However, in the case of the typing algorithm, the soundness and completeness proofs cannot be separated: the inductive proof of one requires the other.

The proof of soundness can be viewed as a mapping from an algorithmic tree to a declarative one. We show that each algorithmic inference rule can be transformed into the corresponding declarative rule, as long as the premises are transformed accordingly, and apply the induction principle.

The soundness requires completeness in the case of $(\textsc{let}^{\textsc{inf}}_{@})$. To prove the soundness, in other words, to transform this rule into its declarative counterpart $(\textsc{let}^{\textsc{inf}}_{@})$, one needs to prove the *principality* of the inferred application type $\uparrow[\widehat{\sigma}]\,Q$. In other words, we need to conclude that any other *declarative* tree infers for the application a supertype of $[\widehat{\sigma}]\,Q$.

The only way to do so is by applying the soundness of minimal instantiation (see the lemma above). However, first, the declarative tree must be converted to the corresponding

algorithmic one (by completeness!). This way, the soundness and completeness proofs are mutually dependent.

Inductive Metric The soundness and completeness lemmas are proven by *mutual* induction. Since the declarative and the algorithmic systems do not depend on each other, we must introduce a uniform *metric* on which the induction is conducted. We define the metric gradually, starting from the auxiliary function—the size of a judgment $\mathsf{size}(J)$.

The size of *declarative* and *algorithmic* judgments is defined as a pair: the first component is the cumulative syntactic size of the terms used in the judgment. The second component depends on the kind of judgment. For regular type inference judgments (such as $\Theta; \Gamma \vdash v \colon P$ or $\Theta; \Gamma \vDash c \colon N$), it is always zero. For application inference judgments ($\Theta \,;\, \Gamma \vdash N \bullet \vec{v} \Longrightarrow\!\!> M$ or $\Theta \,;\, \Gamma \,;\, \Xi \vDash N \bullet \vec{v} \Longrightarrow\!\!> M \dashv \Xi' \,;\, C$), it is equal to the number of prenex quantifiers of the head type N. We need this adjustment to ensure the monotonicity of the metric, since the quantifier instantiation rules $(\forall^{\mathrm{INF}}_{\bullet\Longrightarrow})$ and $(\forall^{\mathrm{INF}}_{\bullet\Longrightarrow})$ only reduce the quantifiers in the head type but they do not change the list of arguments.

Definition (Judgement Size)**.** *For a declarative or an algorithmic typing judgment J, we define a metric* $\mathsf{size}(J)$ *as a pair of integers in the following way:*

- $+$ $\mathsf{size}(\Theta; \Gamma \vdash v \colon P) = (\mathsf{size}(v), 0)$*;*
- $-$ $\mathsf{size}(\Theta; \Gamma \vdash c \colon N) = (\mathsf{size}(c), 0)$*;*
- $\bullet$ $\mathsf{size}(\Theta \,;\, \Gamma \vdash N \bullet \vec{v} \Longrightarrow\!\!> M) = (\mathsf{size}(\vec{v}), \mathsf{npq}(N))$*;*
- $+$ $\mathsf{size}(\Theta; \Gamma \vDash v \colon P) = (\mathsf{size}(v), 0)$*;*
- $-$ $\mathsf{size}(\Theta; \Gamma \vDash c \colon N) = (\mathsf{size}(c), 0)$*;*
- $\bullet$ $\mathsf{size}(\Theta \,;\, \Gamma \,;\, \Xi \vDash N \bullet \vec{v} \Longrightarrow\!\!> M \dashv \Xi' \,;\, C) = (\mathsf{size}(\vec{v}), \mathsf{npq}(N))$.

Here $\mathsf{size}(v)$ *and* $\mathsf{size}(c)$ *is the size of the syntax tree of the term, and* $\mathsf{size}(\vec{v})$ *is the sum of sizes of the terms in* $\vec{v}$*; and* $\mathsf{npq}(N)$ *and* $\mathsf{npq}(P)$ *represent the number of prenex quantifiers, i.e.,*

$$+\ \mathsf{npq}(\exists \overrightarrow{\alpha^{-}}.\, P) = |\overrightarrow{\alpha^{-}}|, \text{ if } P \neq \exists \overrightarrow{\beta^{-}}.\, P',$$
$$-\ \mathsf{npq}(\forall \overrightarrow{\alpha^{+}}.\, N) = |\overrightarrow{\alpha^{+}}|, \text{ if } N \neq \forall \overrightarrow{\beta^{+}}.\, N'.$$

Notice that for *algorithmic* inference system, $\mathsf{size}(J)$ decreases in all the inductive steps, i.e., for each inference rule, the size of the premise judgments is strictly less than the size of the conclusion. However, the *declarative* inference system has rules $(\simeq^{\mathrm{INF}}_{-})$ and $(\simeq^{\mathrm{INF}}_{+})$, that 'step to' an equivalent type, and thus, technically, might keep the judgment unchanged altogether.

To deal with this issue, we introduce the metric on the entire *inference trees* rather than on judgments, and plug into this metric the parameter that certainly decreases in rules $(\simeq^{\mathrm{INF}}_{-})$ and $(\simeq^{\mathrm{INF}}_{+})$—the number of such nodes in the inference tree. We denote this number as $\mathsf{eq\_nodes}(T)$. Then the final metric is defined as a pair in the following way.

Definition (Inference Tree Metric)**.** *For a tree T, inferring a declarative or an algorithmic judgement J, we define* $\mathsf{metric}(T)$ *as follows:*

$$\mathsf{metric}(T) = \begin{cases} (\mathsf{size}(J), \mathsf{eq\_nodes}(T)) & \text{if } J \text{ represents a declarative judgement,} \\ (\mathsf{size}(J), 0) & \text{if } J \text{ represents an algorithmic judgement.} \end{cases}$$

Here $\mathsf{eq\_nodes}(T)$ *is the number of nodes in T labeled with* $(\simeq^{INF}_{+})$ *or* $(\simeq^{INF}_{-})$.

This metric is suitable for mutual induction on the soundness and completeness of the typing algorithm. First, it is monotone w.r.t. the inference rules, and this way, we can always apply the induction hypothesis to premises of each rule. Second, the induction hypothesis is powerful enough, so we can use the completeness of the algorithm in the soundness proof, where required. For instance, to prove the soundness of typing in case of $\Theta; \Gamma \vDash \mathtt{let}\ x = v(\vec{v});\ c' : N$, we can assume, that *completeness* holds for a term of shape $\Theta\,; \Gamma \vdash M \bullet \vec{v} \Longrightarrow K$, since $\mathsf{size}(args) < \mathsf{size}(letx = v(args); c')$. This is exactly what allows us to deal with the case of $(\textsc{let}^{\textsc{inf}}_{@})$, because then we can conclude that the inferred type (of a declarative judgment $\Theta\,; \Gamma \vdash M \bullet \vec{v} \Longrightarrow \uparrow[\widehat{\sigma}]\, Q$ constructed by the induction hypothesis) is unique.

6 Extensions and Future Work

In this section, we discuss several extensions to the system and the algorithm. Some of them can be incorporated into the system immediately, while others are beyond the scope of this work, and thus are left for future research.

6.1 Explicit Type Application

In our system, all type applications are inferred implicitly: the algorithm *Explicit* type application can be added to the declarative system by the following rule:

$$\frac{\Theta; \Gamma \vdash c : \forall \overrightarrow{\alpha^{+}}.\ N}{\Theta; \Gamma \vdash c\{\vec{P}\} : [\vec{P}/\overrightarrow{\alpha^{+}}]N}\ (TApp^{\textsc{inf}})$$

However, this rule alone would cause ambiguity. The declarative system does not fix the order of the quantifiers, which means that $\forall\alpha^{+}.\ \forall\beta^{+}.\ N$ and $\forall\beta^{+}.\ \forall\alpha^{+}.\ N$ can be inferred as a type of c interchangeably. But that would make explicit instantiation $c\{P\}$ ambiguous, as it is unclear whether α^{+} or β^{+} should be instantiated with P.

Solution 1: Declarative Normalization One way to resolve this ambiguity is to fix the order of quantifiers. The algorithm already performs the ordering of the quantifiers in the normalization procedure. This way, we could require the inferred type to be normalized to specify the order of the quantifiers:

$$\frac{\Theta; \Gamma \vdash c : \forall \overrightarrow{\alpha^{+}}.\ N \quad \mathtt{nf}\ (\forall \overrightarrow{\alpha^{+}}.\ N) = \forall \overrightarrow{\alpha^{+}}.\ N}{\Theta; \Gamma \vdash c\{\vec{P}\} : [\vec{P}/\overrightarrow{\alpha^{+}}]N}\ (TApp^{\textsc{inf}})$$

The drawback of this approach is that it adds another point where the internal algorithmic concept—normalization—is exposed to the 'surface' declarative system.

Solution 2: Elementary Type Inference An alternative approach to explicit type application was proposed by (Zhao and d. S. Oliveira, 2022). In this work, the subtyping relation is restricted in such a way that $\forall\alpha^{+}.\ \forall\beta^{+}.\ N$ and $\forall\beta^{+}.\ \forall\alpha^{+}.\ N$ are *not* mutual subtypes (as

long as $\alpha^+ \in \mathtt{fv}\,(N)$ and $\beta^+ \in \mathtt{fv}\,(N)$). It implies that the order of the quantifiers of the inferred type is unique, and thus, the explicit type application is unambiguous.

These restrictions can be incorporated into our system by replacing the polymorphic subtyping rules $(\forall^{\leqslant})$ and $(\exists^{\geqslant})$ with the following stronger versions:

$$\frac{\Theta, \overrightarrow{\beta^+} \vdash \sigma : \overrightarrow{\alpha^+} \quad \Theta, \overrightarrow{\beta^+} \vdash [\sigma]N \leqslant M}{\Theta \vdash \forall\overrightarrow{\beta^+}, \overrightarrow{\alpha^+}.\, N \leqslant \forall\overrightarrow{\beta^+}.\, M}\ (\mathrm{E}\forall^{\leqslant}) \qquad \frac{\Theta, \overrightarrow{\beta^-} \vdash \sigma : \overrightarrow{\alpha^-} \quad \Theta, \overrightarrow{\beta^-} \vdash [\sigma]P \geqslant Q}{\Theta \vdash \exists\overrightarrow{\beta^-}, \overrightarrow{\alpha^-}.\, P \geqslant \exists\overrightarrow{\beta^-}.\, Q}\ (\mathrm{E}\exists^{\geqslant})$$

According to these rules, if two quantified ($\forall$ or $\exists$) types are subtypes of each other, they must have the same top-level quantifiers. Moreover, the equivalence on types (mutual subtyping) degenerates to *equality* up to alpha-conversion.

To accommodate these changes in the *algorithm*, it suffices to (i) replace the normalization procedure with identity: $\mathtt{nf}\,(N) \stackrel{\text{def}}{=} N$, $\mathtt{nf}\,(P) \stackrel{\text{def}}{=} P$, since the new equivalence classes are singletons; (ii) modify the least upper bound polymorphic rule $(\exists^{\vee})$ so that it requires the quantifiers to be equal (and performs alpha-conversion if necessary):

$$\frac{\Theta, \overrightarrow{\alpha^-} \vDash P_1 \vee P_2 = Q \quad \overrightarrow{\alpha^-} \subseteq \mathtt{fv}\,Q}{\Theta \vDash \exists\overrightarrow{\alpha^-}.\, P_1 \vee \exists\overrightarrow{\alpha^-}.\, P_2 = Q}\ (\mathrm{E}\exists^{\vee})$$

(iii) replace the subtyping polymorphic rule $(\forall^{\leqslant})$ by the following rule:

$$\frac{\overrightarrow{\widehat{\alpha^+}}\ \mathtt{are\ fresh} \quad \Theta, \overrightarrow{\beta^+}; \Xi, \overrightarrow{\widehat{\alpha^+}}\{\Theta, \overrightarrow{\beta^+}\} \vDash [\overrightarrow{\widehat{\alpha^+}}/\overrightarrow{\alpha^+}]N \leqslant M \dashv C}{\Theta;\ \Xi \vDash \forall\overrightarrow{\beta^+}, \overrightarrow{\alpha^+}.\, N \leqslant \forall\overrightarrow{\beta^+}.\, M \dashv C}\ (\mathrm{E}\forall^{\leqslant})$$

and (iv) update the existential rule $(\exists^{\geqslant})$ symmetrically.

After these changes, the rule $(TApp^{\mathrm{INF}})$ and its algorithmic counterpart can be used to infer the the type of $c\{P\}$.

Elementary type inference is much more restrictive than what we present. As mentioned, it forbids the quantifier reordering; besides, as soon as the right-hand side of the subtyping relation is polymorphic, it restricts the instantiation of the left-hand side quantifiers, which *disallows*, for example, $\cdot \vdash \forall\alpha^+.\, \uparrow\alpha^+ \leqslant \forall\alpha^+.\, \uparrow\downarrow\uparrow\alpha^+$. On the other hand, the elementary subtyping system can be smoothly extended with bounded quantification, which we discuss later.

Solution 3: labeled quantifiers A compromise solution that resolves the ambiguity of explicit instantiation without fixing the order of quantifiers is by using *labeled* quantifiers. Let us discuss how to introduce them to the *negative* part of the system since the positive subsystem is symmetric.

Each type abstraction $\Lambda l \triangleright \alpha^+.\, c$ must be annotated with a label l. This label propagates to the inferred polymorphic type. This way, each polymorphic quantifier $\forall \overrightarrow{l_i \triangleright \alpha_i^+}^{I}.\, N$ is annotated with a label l_i whose index is taken from the *list* of labels I associated with the group of quantifiers. To instantiate a quantifying variable, one refers to it by its label: $c\{l_j \triangleright P\}$, which is expressed by the following typing rules:

$$\frac{\Theta, \alpha^+; \Gamma \vdash c : N}{\Theta; \Gamma \vdash \Lambda l \triangleright \alpha^+ . c : \forall l \triangleright \alpha^+ . N} (\Lambda^{\text{INF}}) \qquad \frac{\Theta; \Gamma \vdash c : \forall \overrightarrow{l_i \triangleright \alpha_i^{+I}} . N}{\Theta; \Gamma \vdash c\{l_j \triangleright P\} : \forall \overrightarrow{l_i \triangleright \alpha_i^{+I \setminus \{j\}}} . [P/\alpha_j^+]N} (TApp^{\text{INF}})$$

The subtyping rule permits an arbitrary order of the quantifiers. However, similarly to the elementary subtyping, each of the right-hand side quantifiers must have a matching left-hand side quantifier with the same label. These matching quantifiers are synchronously removed (in other words, each variable is abstractly instantiated to itself), and the remaining quantifiers are instantiated as usual by a substitution σ.

$$\frac{\{I\} \subseteq \{J\} \quad \Theta, \overrightarrow{\alpha_i^{+I}} \vdash \sigma : \overrightarrow{\alpha_j^{+J \setminus \{I\}}} \quad \Theta, \overrightarrow{\alpha_i^{+I}} \vdash [\sigma]N \leqslant M}{\Theta \vdash \forall \overrightarrow{l_j \triangleright \alpha_j^{+J}} . N \leqslant \forall \overrightarrow{l_i \triangleright \alpha_i^{+I}} . M} (\forall^{\leqslant})$$

The equivalence (mutual subtyping) in this system allows the quantifiers within the same group to be reordered together with their labels. For instance, $\forall l \triangleright \alpha^+ . \forall m \triangleright \beta^+ . N$ and $\forall m \triangleright \beta^+ . \forall l \triangleright \alpha^+ . N$ are subtypes of each other. However, the right-hand side quantifiers are removed synchronously with the matching left-hand side quantifiers, which means that $\cdot \vdash \forall l \triangleright \alpha^+ . \uparrow \alpha^+ \leqslant \forall l \triangleright \alpha^+ . \uparrow\downarrow\uparrow \alpha^+$ is still not allowed.

6.2 Weakening of the subtyping invariants

In order to make the subtyping relation decidable, we made the subtyping of shifts *invariant*, which manifests in the following rules:

$$\frac{\Theta \vdash P \simeq^{\leqslant} Q}{\Theta \vdash \uparrow P \leqslant \uparrow Q} (\uparrow^{\leqslant}) \qquad \frac{\Theta \vdash N \simeq^{\leqslant} M}{\Theta \vdash \downarrow N \geqslant \downarrow M} (\downarrow^{\geqslant})$$

To make the system more expressive, we can relax these rules. Although making both rules covariant would make the system undecidable, it is possible to make the up-shift subtyping covariant while keeping the down-shift invariant:

$$\frac{\Theta \vdash Q \geqslant P}{\Theta \vdash \uparrow P \leqslant \uparrow Q} (\uparrow^{\leqslant}_{\text{RLX}})$$

However, this change requires an update to the algorithm. As one can expect, the corresponding algorithmic rule $(\uparrow^{\leqslant})$ must be changed accordingly:

$$\frac{\Theta;\ \Xi \vDash Q \leqslant P \dashv C}{\Theta;\ \Xi \vDash \uparrow P \leqslant \uparrow Q \dashv C} (\uparrow^{\leqslant}_{\text{RLX}})$$

Also, notice that now the algorithmic variables can occur on *both* sides of the positive subtyping relation, and thus, a more sophisticated constraint solver is needed. To see what kind of constraints will be generated, let us keep the other algorithmic rules unchanged for a moment and make several observations.

First, notice that the negative quantification variables still only occur at *invariant* positions: the negative variables are generated from the existential quantifiers $\exists \alpha^- . P$, and the switch from a positive to a negative type in the syntax path to α^- in P is 'guarded' by *invariant*

down-shift. It means that in the algorithm, the negative constraint entries can only be equivalence entries ($\widehat{\alpha}^- :\simeq N$) but not subtyping entries.

Second, notice that the positive subtyping is still not dependent on the negative subtyping. It means that any leaf-to-root path in the inference tree is monotone: in the first phase, the negative subtyping rules are applied, and in the second phase, only the positive ones.

The invariant that only the left-hand side of the judgment is algorithmic is not preserved. However, the only rule violating this invariant is ($\uparrow^{\leqslant}_{\text{RLX}}$), and it is placed at the interface between negative subtyping and positive subtyping. It means that the *negative* algorithmic variables (produced by $\exists$ in the positive phase of the inference path) still only occur on the left-hand side of the judgment, and the positive algorithmic variables (produced by $\forall$ in the first phase) can occur either on the left-hand side or on the right-hand side (if the switch from the negative to the positive phase is made by ($\uparrow^{\leqslant}_{\text{RLX}}$)) but not both. This way, the produced entries will be of the following form: (i) $\widehat{\alpha}^+ :\simeq P$, (ii) $\widehat{\alpha}^- :\simeq N$, (iii) $\widehat{\alpha}^+ :\geqslant P$, and (iv) $\widehat{\alpha}^+ :\leqslant P$ where P does not contain positive unification variables.

We do not present the details of the constraint resolution procedure or how it is integrated into the subtyping algorithm, leaving it for future work. However, we believe that the system consisting of the mentioned kinds of constraints is solvable for the following reasons. (i) The equivalence entries $\widehat{\alpha}^+ :\simeq P$ and $\widehat{\alpha}^- :\simeq N$ are resolved by standard first-order pattern unification techniques. (ii) The resolution of supertyping entries ($\widehat{\alpha}^+ :\geqslant P$) is discussed in the original algorithm (Section 4.5). (iii) The new kind of entries $\widehat{\alpha}^+ :\leqslant P$ can be reduced by case analysis. If P is a variable β^+ or a shifted computation $\downarrow N$ then $\widehat{\alpha}^+ :\leqslant P$ is equivalent to $\widehat{\alpha}^+ :\simeq P$; otherwise, $\widehat{\alpha}^+ :\leqslant \exists\overrightarrow{\alpha^-}.\downarrow N$ is equivalently replaced by $\widehat{\alpha}^+ :\simeq [\overrightarrow{\widehat{\alpha}^-}/\overrightarrow{\alpha^-}]\downarrow N$ for freshly created $\overrightarrow{\widehat{\alpha}^-}$.

This way, the algorithm can be extended to support the *covariant* up-shift subtyping. This, for example, enriches the relation with such subtypes as $\cdot \vdash \forall\alpha^+.\,\alpha^+ \to \uparrow\alpha^+ \leqslant \downarrow\uparrow\texttt{Int} \to \uparrow\exists\beta^-.\downarrow\beta^-$, which does not hold in the original system. Moreover, this extension also increases the expressiveness of the *type inference*. Namely, when inferring a type of an annotated let binding $\texttt{let}\,x : P = v(\overrightarrow{v});\ c$, we require $\uparrow P$ to be a supertype of $\uparrow Q$—the resulting type of the application $v(\overrightarrow{v})$. In the relaxed system, it can be replaced with the requirement that P is a supertype of Q (without the up-shift), which is more permissive.

In addition, the refinement of the unification algorithm described above makes possible another improvement to the system—bounded quantification.

6.3 Bounded Quantification

After the weakening of subtyping invariants, it is possible to smoothly extend the *elementary* version of the system (the second solution discussed in Section 6.1) with bounded quantification. In particular, we can add upper bounds to polymorphic $\forall$-quantifiers: $\forall\overrightarrow{\alpha^+ \leqslant P}.\,N$.

The declarative subtyping rules for bounded quantification are as expected: the instantiation of quantified variables must satisfy the corresponding bounds ($\Theta, \overrightarrow{\alpha^+} \vdash Q_i \geqslant [\sigma]\beta_i^+$ holds for each i); and the right-hand side quantifiers must be more restrictive than the matching

left-hand side quantifiers ($\Theta \vdash P_i \geqslant P'_i$ for each i):

$$\frac{\overrightarrow{\Theta \vdash P_i \geqslant P'_i} \quad \Theta, \overrightarrow{\alpha^+} \vdash \sigma : \overrightarrow{\beta^+} \quad \overrightarrow{\Theta, \overrightarrow{\alpha^+} \vdash Q_i \geqslant [\sigma]\beta_i^+} \quad \Theta, \overrightarrow{\alpha^+} \vdash [\sigma]N \leqslant M}{\Theta \vdash \forall \overrightarrow{\alpha^+ \leqslant P}.\, \forall \overrightarrow{\beta^+ \leqslant Q}.\, N \leqslant \forall \overrightarrow{\alpha^+ \leqslant P'}.\, M} \; (\mathrm{E}\forall^{\leqslant})$$

The corresponding algorithmic rule, as before, replaces the polymorphic variables $\beta_1 \ldots$ with fresh algorithmic variables $\widehat{\beta_1}^+ \ldots$, and merges the generated constraints with the constraints given in the bounds $\widehat{\beta_i}^+ :\leqslant Q_i$:

$$\frac{\begin{array}{l}\Theta;\, \cdot \vDash \overrightarrow{P_i \geqslant P'_i} \dashv \cdot \quad \overrightarrow{\widehat{\beta}^+} \;\texttt{are fresh} \\ \Theta, \overrightarrow{\alpha^+};\, \Xi, \overrightarrow{\widehat{\beta}^+} \{\Theta, \overrightarrow{\alpha^+}\} \vDash N \leqslant M \dashv C_0 \quad \Xi \vdash C_0 \,\&\, (\overrightarrow{\widehat{\beta_i}^+ :\leqslant Q_i}) = C\end{array}}{\Theta;\, \Xi \vDash \forall \overrightarrow{\alpha^+ \leqslant P}.\, \forall \overrightarrow{\beta^+ \leqslant Q}.\, N \leqslant \forall \overrightarrow{\alpha^+ \leqslant P'}.\, M \dashv C} \; (\mathrm{E}\forall^{\leqslant})$$

Notice that the algorithmic rule ($\mathrm{E}\forall^{\leqslant}$) requires the bounding types $\overrightarrow{P}$ and $\overrightarrow{Q}$ to be *declarative*. Because of that, the generated constraints have shape $\widehat{\beta_i}^+ :\leqslant Q_i$, i.e., the bounding types do not contain algorithmic variables. This kind of constraints is covered by the resolution procedure described in Section 6.2, and thus, ($\mathrm{E}\forall^{\leqslant}$) fits into the algorithmic system without changes to the constraint solver.

However, in order for this rule to be complete, the declarative system must be restricted as well. To guarantee that no algorithmic variable is introduced in the bounding types, we need to forbid the quantifying variables to occur in the bounding types in the declarative system. For that purpose, we must include this requirement in the type well-formedness (here $\mathtt{fbv}N$ denotes the set of variables freely occurring in the quantifiers of N at any depth):

$$\frac{\overrightarrow{\Theta \vdash P_i} \quad \Theta, \overrightarrow{\alpha^+} \vdash N \quad \overrightarrow{\alpha^+} \cap \mathtt{fbv}N = \emptyset}{\Theta \vdash \forall \overrightarrow{\alpha^+ \leqslant P}.\, N} \; (\forall^{\mathrm{WF}})$$

For brevity, we do not discuss in detail the combination of bounded and unbounded quantification in one system. We believe this update is straightforward: the unbounded quantifiers are treated as the weakest constraints, which are trivially satisfied. Analogously, it is possible to extend the system with *lower* bound $\forall$-quantifiers: $\forall \overrightarrow{\alpha^+ \geqslant P}.\, N$. Then the generated constraints will change to $\widehat{\beta_i}^+ :\geqslant Q_i$, which is also covered by the resolution procedure. However, we do not consider bounded *existential* quantification, because in this case, the constraint resolution would require us to find the *greatest lower bound* of two negative types, which is not well-defined (see Section 4.5).

6.4 Bidirectionalization

The algorithm we provide requires that all lambda functions are annotated. This restriction significantly simplifies the type inference by making the terms uniquely define their types. However, it leads to certain redundancies in typing, in particular, the type of a lambda expression cannot be inferred from the context it is used in: the annotated $((\lambda x.\, \mathtt{return}\, x) : (\mathtt{Int} \to \uparrow \mathtt{Int}))$ does not infer $\mathtt{Int} \to \uparrow \mathtt{Int}$.

The well-known way to incorporate this expressiveness into the system is to make the typing bidirectional (Dunfield and Krishnaswami, 2020). The idea is to split the typing judgment into two kinds:

(i) *Synthesis* judgments are used when the type of a term can be *inferred* based exclusively on the term itself. Syntactically, we denote synthesizing judgments as $\Theta\,;\Gamma \vdash c \Rightarrow N$.

(ii) *Checking* judgments assume that the type of a term is *given* from the context, and it is required to *check* that the given type can be assigned to the term. In checking judgments $\Theta\,;\Gamma \vdash c \Leftarrow N$, the type N is considered as an *input*.

To bidirectionalize the system, each typing rule must be oriented either to synthesis or to checking (or both). In particular, the *annotated* lambda-abstractions are synthesizing, and the *unannotated* ones are checking.

$$\frac{\Theta \vdash P \quad \Theta\,;\Gamma, x : P \vdash c \Rightarrow N}{\Theta\,;\Gamma \vdash \lambda x : P.\, c \Rightarrow P \to N}\ (\lambda^{\Leftarrow}) \qquad \frac{\Theta \vdash P \quad \Theta\,;\Gamma, x : P \vdash c \Leftarrow N}{\Theta\,;\Gamma \vdash \lambda x.\, c \Leftarrow P \to N}\ (\lambda^{\Leftarrow})$$

As common for bidirectional systems with subtyping, we would also need to introduce the *subsumption* rules. They allow us to 'forget' the information about the type by switching to the synthesis mode, as long as the synthesized type is more polymorphic than the checked one:

$$\frac{\Theta\,;\Gamma \vdash c \Rightarrow N \quad \Theta \vdash N \leqslant M}{\Theta\,;\Gamma \vdash c \Leftarrow M}\ (\mathrm{SUB}_-) \qquad \frac{\Theta\,;\Gamma \vdash v \Rightarrow P \quad \Theta \vdash Q \geqslant P}{\Theta\,;\Gamma \vdash v \Leftarrow Q}\ (\mathrm{SUB}_+)$$

Most of the original rules will have two bidirectional counterparts, one for each mode. The premises of the rules are oriented with respect to the conclusion. For instance, we need both checking and inferring versions for `return` v:

$$\frac{\Theta\,;\Gamma \vdash v \Leftrightarrow P}{\Theta\,;\Gamma \vdash \texttt{return}\ v \Leftrightarrow \uparrow P}\ (\mathrm{RET}^{\Leftrightarrow})$$

For some rules, however, it only makes sense to have one mode. As mentioned above, the unannotated lambda-abstraction is always checking. On the other hand, an *annotated* lambda, type-level lambda abstraction, or a variable inference is always synthesizing, since the required type information is already given in the term:

$$\frac{\Theta, \alpha^+\,;\Gamma \vdash c \Rightarrow N}{\Theta\,;\Gamma \vdash \Lambda\alpha^+.\, c \Rightarrow \forall\alpha^+.\, N}\ (\Lambda^{\Leftarrow}) \qquad \frac{x : P \in \Gamma}{\Theta\,;\Gamma \vdash x \Rightarrow P}\ (\mathrm{VAR}^{\Rightarrow})$$

Finally, let us discuss the application inference part of the declarative system. Rules $(\emptyset^{\mathrm{INF}}_{\bullet\Rightarrow})$ and $(\forall^{\mathrm{INF}}_{\bullet\Rightarrow})$ are unchanged: they simply do not have typing premises to be oriented. However, the arrow application rule $(\to^{\mathrm{INF}}_{\bullet\Rightarrow})$ infers the result of the application of an arrow $Q \to N$ to a list of arguments $v, \vec{v}$, and it must make sure that the first argument v is typeable with the expected Q. As we will discuss further, simply checking $\Theta\,;\Gamma \vdash v \Leftarrow Q$ turns out to be too powerful and potentially leads to undecidability. Instead, this check is approximated by the combination of $\Theta\,;\Gamma \vdash v \Rightarrow P$ and $\Theta \vdash Q \geqslant P$.

The Algorithm The algorithmic system is bidirectionalized in a similar way. Each typing premise and the conclusion of the rule are oriented to either 'checking' ($\Leftarrow$) or 'synthesizing' ($\Rightarrow$) mode, with respect to the declarative system.

For brevity, we omit the details of the bidirectional algorithm. Let us discuss one especially important rule—arrow application inference. An intuitive way to bidirectionalize this rule is the following:

$$\frac{\Theta\,;\Gamma \vDash v \Leftarrow Q \dashv C_1 \qquad \Theta\,;\Gamma\,;\Xi \vDash N \bullet \vec{v} \Rightarrow\!\!> M \dashv \Xi'\,;C_2 \qquad \Xi \vdash C_1 \,\&\, C_2 = C}{\Theta\,;\Gamma\,;\Xi \vDash Q \to N \bullet v,\, \vec{v} \Rightarrow\!\!> M \dashv \Xi'\,;C} \,(\to^{\bullet\Rightarrow})$$

However, this rule is too permissive. The judgement $\Theta\,;\Gamma \vDash v \Leftarrow Q \dashv C_1$ requires us to check a term against an *algorithmic* type Q. Further, through the lambda function checking $(\lambda^{\Leftarrow})$ the algorithmic types infiltrate the type context and then through variable inference and subsumption, *both* sides of subtyping. This way, $(\to^{\bullet\Rightarrow})$ compromises the important invariant of the subtyping algorithm: now the same algorithmic variable can occur on *both* sides of the subtyping relation.

We believe that the relaxation of this invariant brings the constraint resolution too close to second-order pattern unification, which is undecidable (Goldfarb, 1981). In particular, the constraint $(\widehat{\alpha}^+ :\geqslant \downarrow\uparrow\widehat{\alpha}^+)$ is solvable with $\widehat{\alpha}^+ = \exists\beta^-.\downarrow\beta^-$, since by $(\exists^{\geqslant})$, the existential β^- can be impredicatively instantiated to $\uparrow\exists\beta^-.\downarrow\beta^-$.

Constraints such as $(\widehat{\alpha}^+ :\geqslant \downarrow\uparrow\widehat{\alpha}^+)$ are not merely hypothetical. It occurs, for example, when we infer the type of the following application:

$$\cdot\,;\cdot \vdash \forall\alpha^+.\downarrow(\alpha^+ \to \uparrow\mathtt{Int}) \to \downarrow\uparrow\alpha^+ \to \uparrow\mathtt{Int} \bullet \{\lambda x.\,\lambda y.\,\mathtt{let}\,y' = x(y);\,\mathtt{return}\,y'\} \Rightarrow\!\!> \uparrow\mathtt{Int}$$

For x to be applicable to y, the type of y—$\downarrow\uparrow\alpha^+$ must be a subtype of α^+—the type expected by x. This way, this inference is possible if and only if $(\widehat{\alpha}^+ :\geqslant \downarrow\uparrow\widehat{\alpha}^+)$ is solvable. Using a similar scheme, we can construct different examples whose resolution significantly relies on second-order pattern unification. Thus, we leave the resolution of this type of constraints beyond the scope of this work.

Instead, we strengthen $(\to^{\bullet\Rightarrow})$ in such a way that it never checks a term against an algorithmic type. Type checking v against Q is replaced by a more restrictive premise—checking that v *synthesizes a subtype* of Q:

$$\frac{\Theta\,;\Gamma \vDash v \Rightarrow P \quad \Theta;\,\Xi \vDash Q \geqslant P \dashv C_1 \qquad \Theta\,;\Gamma\,;\Xi \vDash N \bullet \vec{v} \Rightarrow\!\!> M \dashv \Xi'\,;C_2 \qquad \Xi \vdash C_1 \,\&\, C_2 = C}{\Theta\,;\Gamma\,;\Xi \vDash Q \to N \bullet v,\, \vec{v} \Rightarrow\!\!> M \dashv \Xi'\,;C} \,(\to^{\bullet\Rightarrow})$$

As one can expect, the declarative rule is also changed accordingly. In terms of practicality, this change disallows implicit checking against a *polymorphic* type: $\cdot\,;\cdot \vdash \lambda x.\,\mathtt{return}\,x \Leftarrow \forall\alpha^+.\,\alpha^+ \to \uparrow\alpha^+$ is not allowed, because it would require adding algorithmic $x : \widehat{\alpha}^+$ to the context. However, once the instantiation is made explicit or the lambda is explicitly polymorphic, the checking is allowed: $\cdot\,;\cdot \vdash \lambda x.\,\mathtt{return}\,x \Leftarrow \mathtt{Int} \to \uparrow\mathtt{Int}$ and $\cdot\,;\cdot \vdash \Lambda\alpha^+.\,\lambda x : \alpha^+.\,\mathtt{return}\,x \Leftarrow \forall\alpha^+.\,\alpha^+ \to \uparrow\alpha^+$ are both valid.

7 Conclusion

We have presented a type inference algorithm for an impredicative polymorphic lambda calculus with existential types and subtyping. The system is designed in the call-by-push-value paradigm, which allowed us to restrict it to a decidable fragment of the language described declaratively. We presented the inference algorithm and proved its soundness and completeness with respect to the specification. The algorithm combines unification—a standard type inference technique—with anti-unification—which has not been used in type inference before.

References

Chrzaszcz, J. (1998) Polymorphic subtyping without distributivity. Proceedings of the 23rd International Symposium on Mathematical Foundations of Computer Science. Berlin, Heidelberg. Springer-Verlag. pp. 346–355.

Damas, L. & Milner, R. (1982) Principal type-schemes for functional programs. ACM-SIGACT Symposium on Principles of Programming Languages.

Dunfield, J. & Krishnaswami, N. (2020) *Bidirectional Typing*.

Dunfield, J. & Krishnaswami, N. R. (2016) Sound and complete bidirectional typechecking for higher-rank polymorphism with existentials and indexed types. *Proceedings of the ACM on Programming Languages*. **3**, 1 – 28.

Eisenberg, R. A., Duboc, G., Weirich, S. & Lee, D. K. (2021) An existential crisis resolved: type inference for first-class existential types. *Proceedings of the ACM on Programming Languages*. **5**, 1 – 29.

Girard, J.-Y. (1971) Une extension de ĽInterpretation de Gödel à ĽAnalyse, et son application à ĽElimination des coupures dans ĽAnalyse et la theorie des types. In *Proceedings of the Second Scandinavian Logic Symposium*, Fenstad, J. (eds). vol. 63 of *Studies in Logic and the Foundations of Mathematics*. Elsevier. North Holland. pp. 63–92.

Goldfarb, W. D. (1981) The undecidability of the second-order unification problem. *Theor. Comput. Sci.* **13**, 225–230.

Hindley, R. (1969) The principal type-scheme of an object in combinatory logic. *Transactions of the American Mathematical Society*. **146**, 29–60.

Läufer, K. & Odersky, M. (1994) Polymorphic type inference and abstract data types. *ACM Trans. Program. Lang. Syst.* **16**, 1411–1430.

Leijen, D. (2006) First-class polymorphism with existential types.

Levy, P. B. (2006) Call-by-push-value: Decomposing call-by-value and call-by-name. *Higher-Order and Symbolic Computation*. **19**(4), 377–414.

Miller, D. (1991) A logic programming language with lambda-abstraction, function variables, and simple unification. *J. Log. Comput.* **1**(4), 497–536.

Milner, R. (1978) A theory of type polymorphism in programming. *J. Comput. Syst. Sci.* **17**, 348–375.

Pierce, B. & Turner, D. (2000) Local Type Inference. *ACM Transactions on Programming Languages and Systems*. **22**(1), 1–44.

Plotkin, G. D. (1970) A note on inductive generalization. *Machine Intelligence*. **5**(1), 153–163.

Reynolds, J. C. (1970) Transformational systems and the algebraic structure of atomic formulas. pp. 135–151.

Reynolds, J. C. (1974) Towards a theory of type structure. Programming Symposium. Paris, France. Springer. Springer. pp. 408–425.

Serrano, A., Hage, J., Peyton Jones, S. & Vytiniotis, D. (2020) A quick look at impredicativity. *Proceedings of the ACM on Programming Languages*. **4**(ICFP), 1–29.

Tiuryn, J. (1995) Equational axiomatization of bicoercibility for polymorphic types. Foundations of Software Technology and Theoretical Computer Science, 15th Conference, Bangalore, India,

December 18-20, 1995, Proceedings. Springer. pp. 166–179.

Tiuryn, J. & Urzyczyn, P. (1996) The subtyping problem for second-order types is undecidable. Proceedings of the 11th Annual IEEE Symposium on Logic in Computer Science. USA. IEEE Computer Society. p. 74.

Zhao, J. & d. S. Oliveira, B. C. (2022) Elementary type inference. 36th European Conference on Object-Oriented Programming, ECOOP 2022, June 6-10, 2022, Berlin, Germany. Schloss Dagstuhl - Leibniz-Zentrum für Informatik. pp. 2:1–2:28.

Appendix A Definitions and Algorithms

A.1 Declarative Types

A.1.1 Grammar

We assume that there is an infinite set of positive and negative *type* variables. Positive type variables are denoted as α^+, β^+, γ^+, etc. Negative type variables are denoted as α^-, β^-, γ^-, etc. We assume there is an infinite set of *term* variables, which are denoted as x, y, z, etc. A list of objects (variables, types or terms) is denoted by an overline arrow. For instance, $\overrightarrow{\alpha^+}$ is a list of positive type variables, $\overrightarrow{\beta^-}$ is a list of negative type variables, $\overrightarrow{v}$ is a list of values, which are arguments of a function. $\mathtt{fv}\,(P)$ and $\mathtt{fv}\,(N)$ denote the set of free variables in a type P and N, respectively.

Definition 1 (Declarative Types)**.**

Negative declarative types

$$\begin{array}{lll} N,\ M,\ K & ::= & \\ & \mid & \alpha^- \\ & \mid & \uparrow P \\ & \mid & P \to N \\ & \mid & \forall \overrightarrow{\alpha^+}.\ N \end{array}$$

Positive declarative types

$$\begin{array}{lll} P,\ Q,\ R & ::= & \\ & \mid & \alpha^+ \\ & \mid & \downarrow N \\ & \mid & \exists \overrightarrow{\alpha^-}.\ P \end{array}$$

A.1.2 Equalities

For simplicity, we assume alpha-equivalent terms are equal. This way, we assume that substitutions do not capture bound variables. Besides, we equate $\forall \overrightarrow{\alpha^+}.\ \forall \overrightarrow{\beta^+}.\ N$ with $\forall \overrightarrow{\alpha^+},\ \overrightarrow{\beta^+}.\ N$, as well as $\exists \overrightarrow{\alpha^-}.\ \exists \overrightarrow{\beta^-}.\ P$ with $\exists \overrightarrow{\alpha^-},\ \overrightarrow{\beta^-}.\ P$, and lift these equations transitively and congruently to the whole system.

A.1.3 Contexts and Well-formedness

Definition 2 (Declarative Type Context)**.**
Declarative type context Θ *is represented by a set of type variables. The concatenation* Θ_1, Θ_2 *means the union of two contexts* $\Theta_1 \cup \Theta_2$.

$\Theta \vdash P$ and $\Theta \vdash N$ denote that the type is well-formed in the context Θ, which means that each free type variable of the type is contained in Θ (it will be shown later in Lemmas 3 and 4).

Notice that checking the well-formedness of a type is an *algorithmic* procedure, in which both the context and the type are considered inputs. In other words, it is syntax-directed and mode-correct (according to (**?**)), which means that checking the well-formedness of a type can be done recursively by a deterministic algorithm. We will use the well-formedness checking in the inference algorithm, for example, to check that the existential variables do not escape their scope.

Algorithm 1 (Type Well-formedness)**.**

$\boxed{\Theta \vdash N}$ *Negative type well-formedness* $\qquad$ $\boxed{\Theta \vdash P}$ *Positive type well-formedness*

$$\frac{\alpha^- \in \Theta}{\Theta \vdash \alpha^-}\ (VAR_-^{WF}) \qquad \frac{\alpha^+ \in \Theta}{\Theta \vdash \alpha^+}\ (VAR_+^{WF})$$

$$\frac{\Theta \vdash P}{\Theta \vdash \uparrow P}\ (\uparrow^{WF}) \qquad \frac{\Theta \vdash N}{\Theta \vdash \downarrow N}\ (\downarrow^{WF})$$

$$\frac{\Theta \vdash P \quad \Theta \vdash N}{\Theta \vdash P \to N}\ (\to^{WF}) \qquad \frac{\Theta, \overrightarrow{\alpha^-} \vdash P}{\Theta \vdash \exists \overrightarrow{\alpha^-}.\ P}\ (\exists^{WF})$$

$$\frac{\Theta, \overrightarrow{\alpha^+} \vdash N}{\Theta \vdash \forall \overrightarrow{\alpha^+}.\ N}\ (\forall^{WF})$$

A.1.4 Substitutions

Definition 3 (Substitution)**.** *Substitutions (denoted as σ) are represented by total functions from variables to types, preserving the polarity.*

Algorithm 2 (Substitution Application)**.** *Substitution application is denoted as $[\sigma]P$ and $[\sigma]N$. It is defined naturally as follows:*

$[\sigma]\alpha^+ = \sigma(\alpha^+)$ $\qquad$ $[\sigma](P \to N) = [\sigma]P \to [\sigma]N$

$[\sigma]\alpha^- = \sigma(\alpha^-)$ $\qquad$ $[\sigma]\exists\overrightarrow{\alpha^-}.\ Q = \exists\overrightarrow{\alpha^-}.\ [\sigma]Q$

$[\sigma]\downarrow N = \downarrow[\sigma]N$ $\qquad$ $[\sigma]\forall\overrightarrow{\alpha^+}.\ N = \forall\overrightarrow{\alpha^+}.\ [\sigma]N$ *(assuming the variable capture never happens)*

$[\sigma]\uparrow P = \uparrow[\sigma]P$

Definition 4 (Substitution Signature)**.** *The signature $\Theta' \vdash \sigma : \Theta$ means that*

1. *for any $\alpha^\pm \in \Theta$, $\Theta' \vdash [\sigma]\alpha^\pm$; and*
2. *for any $\alpha^\pm \notin \Theta'$, $[\sigma]\alpha^\pm = \alpha^\pm$.*

A substitution can be restricted to a set of variables. The restricted substitution is define as expected.

Definition 5 (Subsitution Restriction)**.** *The specification $\sigma|_{vars}$ is defined as a function such that*

1. *$\sigma|_{vars}(\alpha^\pm) = \sigma(\alpha^\pm)$, if $\alpha^\pm \in vars$; and*
2. *$\sigma|_{vars}(\alpha^\pm) = \alpha^\pm$, if $\alpha^\pm \notin vars$.*

Two substitutions can be composed in two ways: $\sigma_2 \circ \sigma_1$ corresponds to a consecutive application of σ_1 and σ_2, while $\sigma_2 \lll \sigma_1$ depends on a signature of σ_1 and modifies σ_1 by applying σ_2 to its results on the domain.

Definition 6 (Substitution Composition)**.** *$\sigma_2 \circ \sigma_1$ is defined as a function such that $\sigma_2 \circ \sigma_1(\alpha^\pm) = \sigma_2(\sigma_1(\alpha^\pm))$.*

Definition 7 (Monadic Substitution Composition)**.** *Suppose that $\Theta' \vdash \sigma_1 : \Theta$. Then we define $\sigma_2 \lll \sigma_1$ as $(\sigma_2 \circ \sigma_1)|_\Theta$.*

Notice that the result of $\sigma_2 \ll \sigma_1$ depends on the specification of σ_1, which is not unique. However, we assume that the used specification clear from the context of the proof.

Definition 8 (Equivalent Substitutions)**.** *The substitution equivalence judgement* $\Theta' \vdash \sigma_1 \simeq^{\leqslant} \sigma_2 : \Theta$ *indicates that on the domain* Θ*, the result of* σ_1 *and* σ_2 *are equivalent in context* Θ'*. Formally, for any* $\alpha^{\pm} \in \Theta$, $\Theta' \vdash [\sigma_1]\alpha^{\pm} \simeq^{\leqslant} [\sigma_2]\alpha^{\pm}$.

Sometimes it is convenient to construct substitution explicitly mapping each variable from a list (or a set) to a type. Such substitutions are denoted as $\vec{P}/\overrightarrow{\alpha^+}$ and $\vec{N}/\overrightarrow{\alpha^-}$, where $\vec{P}$ and $\vec{N}$ are lists of the corresponding types.

Definition 9 (Explicit Substitution)**.**

– *Suppose that* $\overrightarrow{\alpha^-}$ *is a list of negative type variables, and* $\vec{N}$ *is a list of negative types of the same length. Then* $\vec{N}/\overrightarrow{\alpha^-}$ *denotes a substitution such that*

1. *for* $\alpha_i^+ \in \overrightarrow{\alpha^-}$, $[\vec{N}/\overrightarrow{\alpha^-}]\alpha_i^+ = N_i$;
2. *for* $\beta^+ \notin \overrightarrow{\alpha^-}$, $[\vec{N}/\overrightarrow{\alpha^-}]\beta^+ = \beta^+$.

\+ *Positive explicit substitution* $\vec{P}/\overrightarrow{\alpha^+}$ *is defined symmetrically.*

A.1.5 Declarative Subtyping

Subtyping is one of the key mechanisms of our system. It realizes the polymorphism: abstract $\forall$ and $\exists$ types can be used where concrete types are expected, exactly because of the subtyping relation between them.

Definition 10.

$\boxed{\Theta \vdash N \leqslant M}$ *Negative subtyping*

$$\frac{}{\Theta \vdash \alpha^- \leqslant \alpha^-}\ (\textsc{Var}^{\leqslant}_{-})$$

$$\frac{\Theta \vdash P \simeq^{\leqslant} Q}{\Theta \vdash \uparrow P \leqslant \uparrow Q}\ (\uparrow^{\leqslant})$$

$$\frac{\Theta \vdash P \geqslant Q \quad \Theta \vdash N \leqslant M}{\Theta \vdash P \to N \leqslant Q \to M}\ (\to^{\leqslant})$$

$$\frac{\Theta, \overrightarrow{\beta^+} \vdash \sigma : \overrightarrow{\alpha^+} \quad \Theta, \overrightarrow{\beta^+} \vdash [\sigma]N \leqslant M}{\Theta \vdash \forall\overrightarrow{\alpha^+}.\, N \leqslant \forall\overrightarrow{\beta^+}.\, M}\ (\forall^{\leqslant})$$

$\boxed{\Theta \vdash P \geqslant Q}$ *Positive supertyping*

$$\frac{}{\Theta \vdash \alpha^+ \geqslant \alpha^+}\ (\textsc{Var}^{\geqslant}_{+})$$

$$\frac{\Theta \vdash N \simeq^{\leqslant} M}{\Theta \vdash \downarrow N \geqslant \downarrow M}\ (\downarrow^{\geqslant})$$

$$\frac{\Theta, \overrightarrow{\beta^-} \vdash \sigma : \overrightarrow{\alpha^-} \quad \Theta, \overrightarrow{\beta^-} \vdash [\sigma]P \geqslant Q}{\Theta \vdash \exists\overrightarrow{\alpha^-}.\, P \geqslant \exists\overrightarrow{\beta^-}.\, Q}\ (\exists^{\geqslant})$$

$\boxed{\Theta \vdash N \simeq^{\leqslant} M}$ *Negative equivalence*

$$\frac{\Theta \vdash N \leqslant M \quad \Theta \vdash M \leqslant N}{\Theta \vdash N \simeq^{\leqslant} M}\ (\simeq^{\leqslant}_{-})$$

$\boxed{\Theta \vdash P \simeq^{\leqslant} Q}$ *Positive equivalence*

$$\frac{\Theta \vdash P \geqslant Q \quad \Theta \vdash Q \geqslant P}{\Theta \vdash P \simeq^{\leqslant} Q}\ (\simeq^{\leqslant}_{+})$$

The following observations about the declarative subtyping are worth noting:

- $(\textsc{Var}^{\leqslant}_{-})$ and $(\textsc{Var}^{\geqslant}_{+})$ make the subtyping reflexive on variables (and further, on any type).

- $(\to^{\leqslant})$ is standard: the arrow is covariant on the resulting type and contravariant on the argument type.
- $(\downarrow^{\geqslant})$ and $(\uparrow^{\leqslant})$ are non-standard: the subtyping is *invariant* for shifts. This way, the subtyping of shifted types in one direction implies the subtyping in the opposite direction. Although this rule restricts the subtyping relation, it makes the system decidable.
- $(\forall^{\leqslant})$ and $(\exists^{\geqslant})$ are the only non-algorithmic rules: the substitution for the quantified variable is not specified, those, these rules 'drive' the subtyping relation.

In the next section, we present the sound and complete algorithm checking whether one type is a subtype of another according to Definition 10.

A.2 Algorithmic Types

A.2.1 Grammar

In the algorithmic system, we extend the grammar of types by adding positive and negative *algorithmic variables* ($\widehat{\alpha}^{+}$, $\widehat{\beta}^{+}$, $\widehat{\gamma}^{+}$, etc. and $\widehat{\alpha}^{-}$, $\widehat{\beta}^{-}$, $\widehat{\gamma}^{-}$, etc.). They represent the unknown types, which will be inferred by the algorithm. This way, we add two base cases to the grammar of positive and negative types and use highlight to denote that the type can potentially contain algorithmic variables.

Definition 11 (Algorithmic Types)**.**

Negative algorithmic type		*Positive algorithmic type*	
$N,\ M$	$::=$	$P,\ Q$	$::=$
	$\mid\ \widehat{\alpha}^{-}$		$\mid\ \widehat{\alpha}^{+}$
	$\mid\ \alpha^{-}$		$\mid\ \alpha^{+}$
	$\mid\ \uparrow P$		$\mid\ \downarrow N$
	$\mid\ P \to N$		$\mid\ \exists\overrightarrow{\alpha^{-}}.\ P$
	$\mid\ \forall\overrightarrow{\alpha^{+}}.\ N$		

A.2.2 Fresh Variable Selection

Both the subtyping and the type inference algorithm rely on the ability to select fresh, unused variables. For a set of variables *vars*, it is indicated as *vars* `are fresh` in the inference rules. We assume that the selection subroutine always succeeds and is deterministic. In other words, whenever it is called in an algorithmic inference rule, it returns the same result, uniquely determined by the input of this rule.

A.2.3 Variable Algorithmization

In several places of our algorithm, in particular, during algorithmic subtyping, we turn a declarative type into an algorithmic one via replacing certain type variables with fresh algorithmic variables. We call this procedure *variable algorithmization*, and define it as follows.

Definition 12 (Variable Algorithmization)**.** *Suppose that $\overrightarrow{\alpha^{-}}$ is a list of negative type variables and $\overrightarrow{\widehat{\alpha}^{-}}$ is a list of negative algorithmic variables of the same length. Then $\overrightarrow{\widehat{\alpha}^{-}}/\overrightarrow{\alpha^{-}}$ is a substitution-like procedure replacing each $\alpha_i^{-} \in \overrightarrow{\alpha^{-}}$ in a type for $\widehat{\alpha_i}^{-} \in \overrightarrow{\widehat{\alpha}^{-}}$.*

Conversely, we have the opposite procedure turning algorithmic type variables into declarative type variables via *dealgorithmization*.

Definition 13 (Variable Dealgorithmization)**.** *Suppose that* $\overrightarrow{\widehat{\alpha}^{-}}$ *is a list of negative algorithmic variables and* $\overrightarrow{\alpha^{-}}$ *is a list of negative type variables of the same length. Then* $\overrightarrow{\alpha^{-}}/\overrightarrow{\widehat{\alpha}^{-}}$ *is a substitution-like procedure replacing each* $\widehat{\alpha}_i^{-} \in \overrightarrow{\widehat{\alpha}^{-}}$ *in a type for* $\alpha_i^{-} \in \overrightarrow{\alpha^{-}}$.

A.2.4 Contexts and Well-formedness

Definition 14 (Algorithmic Type Context $\widehat{\Theta}$)**.**
Algorithmic type context $\widehat{\Theta}$ *is represented by a set of* algorithmic *type variables* ($\widehat{\alpha}^{+}$, $\widehat{\alpha}^{-}$, $\widehat{\beta}^{+}$, ...). *The concatenation* $\widehat{\Theta}_1, \widehat{\Theta}_2$ *means the union of two contexts* $\widehat{\Theta}_1 \cup \widehat{\Theta}_2$.

$\Theta\,;\widehat{\Theta} \vdash P$ and $\Theta\,;\widehat{\Theta} \vdash N$ are used to denote that the algorithmic type is well-formed in the contexts Θ and $\widehat{\Theta}$, which means that each algorithmic variable of the type is contained in $\widehat{\Theta}$, and each free declarative type variable of the type is contained in Θ.

Algorithm 3 (Algorithmic Type Well-formedness)**.**

$\boxed{\Theta\,;\widehat{\Theta} \vdash N}$ *Negative type well-formedness* $\qquad$ $\boxed{\Theta\,;\widehat{\Theta} \vdash P}$ *Positive type well-formedness*

$$\frac{\alpha^{-} \in \Theta}{\Theta\,;\widehat{\Theta} \vdash \alpha^{-}}\,(VAR_{-}^{WF}) \qquad \frac{\alpha^{+} \in \Theta}{\Theta\,;\widehat{\Theta} \vdash \alpha^{+}}\,(VAR_{+}^{WF})$$

$$\frac{\widehat{\alpha}^{-} \in \widehat{\Theta}}{\Theta\,;\widehat{\Theta} \vdash \widehat{\alpha}^{-}}\,(UVAR_{-}^{WF}) \qquad \frac{\widehat{\alpha}^{+} \in \widehat{\Theta}}{\Theta\,;\widehat{\Theta} \vdash \widehat{\alpha}^{+}}\,(UVAR_{+}^{WF})$$

$$\frac{\Theta\,;\widehat{\Theta} \vdash P}{\Theta\,;\widehat{\Theta} \vdash \uparrow P}\,(\uparrow^{WF}) \qquad \frac{\Theta\,;\widehat{\Theta} \vdash N}{\Theta\,;\widehat{\Theta} \vdash \downarrow N}\,(\downarrow^{WF})$$

$$\frac{\Theta\,;\widehat{\Theta} \vdash P \quad \Theta\,;\widehat{\Theta} \vdash N}{\Theta\,;\widehat{\Theta} \vdash P \to N}\,(\to^{WF}) \qquad \frac{\Theta, \overrightarrow{\alpha^{+}}\,;\widehat{\Theta} \vdash P}{\Theta\,;\widehat{\Theta} \vdash \exists \overrightarrow{\alpha^{+}}.\, P}\,(\exists^{WF})$$

$$\frac{\Theta, \overrightarrow{\alpha^{+}}\,;\widehat{\Theta} \vdash N}{\Theta\,;\widehat{\Theta} \vdash \forall \overrightarrow{\alpha^{+}}.\, N}\,(\forall^{WF})$$

Algorithmic Type Context are used in the unification algorithm. In the subtyping algorithm, the context needs to remember additional information. In the subtyping context, each algorithmic variable is associated with a context it must be instantiated in (i.e. the context in which the type replacing the variable must be well-formed). This association is represented by *algorithmic subtyping context* Ξ.

Definition 15 (Algorithmic Subtyping Context Ξ)**.**
Algorithmic Subtyping Context Ξ *is represented by a set of entries of form* $\widehat{\alpha}^{+}\{\Theta\}$ *and* $\widehat{\alpha}^{-}\{\Theta\}$, *where* $\widehat{\alpha}^{+}$ *and* $\widehat{\alpha}^{-}$ *are algorithmic variables, and* Θ *is a context in which they must be instantiated. We assume that no two entries associating the same variable appear in* Ξ.

$\mathtt{dom}\,(\Xi)$ *denotes the set of variables appearing in* Ξ: $\mathtt{dom}\,(\Xi) = \{\widehat{\alpha}^{\pm} \mid \widehat{\alpha}^{\pm}\{\Theta\} \in \Xi\}$. *If* $\widehat{\alpha}^{\pm}\{\Theta\} \in \Xi$, *we denote* Θ *as* $\Xi(\widehat{\alpha}^{\pm})$.

A.2.5 Subsitutions

A substitution that operates on algorithmic type variables is denoted as $\widehat{\sigma}$. It is defined as a total function from algorithmic type variables to *declarative* types, preserving the polarity.

The signature $\Xi \vdash \widehat{\sigma} : \widehat{\Theta}$ means that $\widehat{\Theta} \subseteq \mathtt{dom}\,(\Xi)$ and $\widehat{\sigma}$ maps each algorithmic variable from $\widehat{\Theta}$ to a type well-formed in $\Xi(\widehat{\alpha}^{\pm})$; and for each variable not appearing in $\mathtt{dom}\,(\Xi)$, it acts as identity.

Definition 16 (Signature of Algorithmic Substitution)**.**

- $\Xi \vdash \widehat{\sigma} : \widehat{\Theta}$ *means that*
  1. *for any* $\widehat{\alpha}^{\pm} \in \widehat{\Theta}$*, there exists* Θ *such that* $\widehat{\alpha}^{\pm}\{\Theta\} \in \Xi$ *and* $\Theta \vdash [\widehat{\sigma}]\widehat{\alpha}^{\pm}$*;*
  2. *for any* $\widehat{\alpha}^{\pm} \notin \widehat{\Theta}$*,* $[\widehat{\sigma}]\widehat{\alpha}^{\pm} = \widehat{\alpha}^{\pm}$.
- $\Theta \vdash \widehat{\sigma} : \widehat{\Theta}$ *means that*
  1. *for any* $\widehat{\alpha}^{\pm} \in \widehat{\Theta}$*,* $\Theta \vdash [\widehat{\sigma}]\widehat{\alpha}^{\pm}$*;*
  2. *for any* $\widehat{\alpha}^{\pm} \notin \widehat{\Theta}$*,* $[\widehat{\sigma}]\widehat{\alpha}^{\pm} = \widehat{\alpha}^{\pm}$.

In the anti-unification algorithm, we use another kind of substitution. In contrast to algorithmic substitution $\widehat{\sigma}$, it allows mapping algorithmic variables to *algorithmic* types. Additionally, anti-unification substitution is restricted to the *negative* segment of the language. Anti-unification substitution is denoted as $\widehat{\tau}$ and $\widehat{\rho}$.a

The pair of contexts Θ and $\widehat{\Theta}$, in which the results of an anti-unification substitution are formed, is fixed for this substitution. This way, $\Theta\,;\widehat{\Theta}_2 \vdash \widehat{\tau} : \widehat{\Theta}_1$ means that $\widehat{\tau}$ maps each negative algorithmic variable appearing in $\widehat{\Theta}_1$ to a term well-formed in Θ and $\widehat{\Theta}_2$.

Definition 17 (Signature of Anti-unification substitution)**.** $\Theta\,;\widehat{\Theta}_2 \vdash \widehat{\tau} : \widehat{\Theta}_1$ *means that*

1. *for any* $\widehat{\alpha}^{-} \in \widehat{\Theta}_1$*,* $\Theta\,;\widehat{\Theta}_2 \vdash [\widehat{\tau}]\widehat{\alpha}^{-}$ *and*
2. *for any* $\widehat{\alpha}^{-} \notin \widehat{\Theta}_1$*,* $[\widehat{\tau}]\widehat{\alpha}^{-} = \widehat{\alpha}^{-}$.

A.2.6 Equivalence and Normalization

The subtyping-induced equivalence (Definition 10) is non-trivial: there are types that are subtypes of each other but not equal. For example, $\forall\alpha^{+}, \beta^{+}.\, \alpha^{+} \to \uparrow\beta^{+}$ is a subtype and a supertype of $\forall\alpha^{+}, \beta^{+}.\, \beta^{+} \to \uparrow\alpha^{+}$ and of, for example, $\forall\alpha^{+}, \beta^{+}.\, \beta^{+} \to \uparrow\exists\gamma^{-}.\, \alpha^{+}$, although these types are not alpha-equivalent. For the subtyping algorithm, it is crucial to be able to check whether two types are equivalent, without checking mutual subtyping. For this purpose we define the normalization procedure, which allows us to uniformly choose the representative type of the equivalence class. This way, the equivalence checking is reduced to normalization and equality checking.

For clarification of the proofs and better understanding of the system, we introduce an intermediate relation—*declarative equivalence*. As will be shown in Lemmas 29 and 34, this relation is equivalent to the subtyping-induced equivalence, but does not depend on it. Although this relation is not defined algorithmically, it gives the intuition of what types our system considers equivalent. Specifically, in addition to *alpha-equivalence*, our

system allows for *reordering of adjacent quantifiers*, and *introduction/elimination of unused quantifiers*.

The non-trivial rules of the declarative equivalence are $(\forall^{\simeq^D})$ and $(\exists^{\simeq^D})$. Intuitively, the variable bijection μ reorders the quantifiers before the recursive call on the body of the quantified type. It will be covered formally in Section C.1.4.

Definition 18 (Declarative Type Equivalence)**.**

$\boxed{N \simeq^D M}$ *Negative type equivalence* $\qquad$ $\boxed{P \simeq^D Q}$ *Positive type equivalence*

$$\frac{}{\alpha^- \simeq^D \alpha^-}\,(\textsc{Var}_-^{\simeq^D}) \qquad \frac{}{\alpha^+ \simeq^D \alpha^+}\,(\textsc{Var}_+^{\simeq^D})$$

$$\frac{P \simeq^D Q}{\uparrow P \simeq^D \uparrow Q}\,(\uparrow^{\simeq^D}) \qquad \frac{N \simeq^D M}{\downarrow N \simeq^D \downarrow M}\,(\downarrow^{\simeq^D})$$

$$\frac{P \simeq^D Q \quad N \simeq^D M}{P \to N \simeq^D Q \to M}\,(\to^{\simeq^D}) \qquad \frac{\begin{array}{c}\mu : (\overrightarrow{\beta^-} \cap \mathtt{fv}\, Q) \leftrightarrow (\overrightarrow{\alpha^-} \cap \mathtt{fv}\, P) \\ \overrightarrow{\alpha^-} \cap \mathtt{fv}\, Q = \emptyset \quad P \simeq^D [\mu] Q\end{array}}{\exists \overrightarrow{\alpha^-}.\, P \simeq^D \exists \overrightarrow{\beta^-}.\, Q}\,(\exists^{\simeq^D})$$

$$\frac{\begin{array}{c}\mu : (\overrightarrow{\beta^+} \cap \mathtt{fv}\, M) \leftrightarrow (\overrightarrow{\alpha^+} \cap \mathtt{fv}\, N) \\ \overrightarrow{\alpha^+} \cap \mathtt{fv}\, M = \emptyset \quad N \simeq^D [\mu] M\end{array}}{\forall \overrightarrow{\alpha^+}.\, N \simeq^D \forall \overrightarrow{\beta^+}.\, M}\,(\forall^{\simeq^D})$$

As the equivalence includes arbitrary reordering of quantified variables, the normalization procedure is needed to choose the canonical order. For this purpose, we introduce an auxiliary procedure—variable ordering. Intuitively, $\mathtt{ord}\ \mathit{vars}\ \mathtt{in}\ N$ returns a list of variables from vars in the order they appear in N.

Algorithm 4 (Variable Ordering)**.**

$\boxed{\mathtt{ord}\ \mathit{vars}\ \mathtt{in}\ N = \overrightarrow{\alpha}}$ *variable ordering in a negative type*

$$\frac{\alpha^- \in \mathit{vars}}{\mathtt{ord}\ \mathit{vars}\ \mathtt{in}\ \alpha^- = \alpha^-}\,(\textsc{Var}_{-\in}^{\textsc{Ord}}) \qquad \frac{\alpha^- \notin \mathit{vars}}{\mathtt{ord}\ \mathit{vars}\ \mathtt{in}\ \alpha^- = \cdot}\,(\textsc{Var}_{-\notin}^{\textsc{Ord}})$$

$$\frac{\mathtt{ord}\ \mathit{vars}\ \mathtt{in}\ P = \overrightarrow{\alpha}}{\mathtt{ord}\ \mathit{vars}\ \mathtt{in} \uparrow P = \overrightarrow{\alpha}}\,(\uparrow^{\textsc{Ord}})$$

$$\frac{\mathtt{ord}\ \mathit{vars}\ \mathtt{in}\ P = \overrightarrow{\alpha}_1 \quad \mathtt{ord}\ \mathit{vars}\ \mathtt{in}\ N = \overrightarrow{\alpha}_2}{\mathtt{ord}\ \mathit{vars}\ \mathtt{in}\ P \to N = \overrightarrow{\alpha}_1, (\overrightarrow{\alpha}_2 \setminus \overrightarrow{\alpha}_1)}\,(\to^{\textsc{Ord}})$$

$$\frac{\mathit{vars} \cap \overrightarrow{\alpha^+} = \emptyset \quad \mathtt{ord}\ \mathit{vars}\ \mathtt{in}\ N = \overrightarrow{\alpha}}{\mathtt{ord}\ \mathit{vars}\ \mathtt{in} \forall \overrightarrow{\alpha^+}.\, N = \overrightarrow{\alpha}}\,(\forall^{\textsc{Ord}})$$

$\boxed{\mathtt{ord}\ \mathit{vars}\ \mathtt{in}\ P = \overrightarrow{\alpha}}$

variable ordering in a positive type

$$\frac{\alpha^+ \in \mathit{vars}}{\mathtt{ord}\ \mathit{vars}\ \mathtt{in}\ \alpha^+ = \alpha^+}\,(\textsc{Var}_{+\in}^{\textsc{Ord}})$$

$$\frac{\alpha^+ \notin \mathit{vars}}{\mathtt{ord}\ \mathit{vars}\ \mathtt{in}\ \alpha^+ = \cdot}\ (\mathit{VAR}^{\mathit{ORD}}_{+\notin})$$

$$\frac{\mathtt{ord}\ \mathit{vars}\ \mathtt{in}\ N = \overrightarrow{\alpha}}{\mathtt{ord}\ \mathit{vars}\ \mathtt{in} \downarrow N = \overrightarrow{\alpha}}\ (\downarrow^{\mathit{ORD}})$$

$$\frac{\mathit{vars} \cap \overrightarrow{\alpha^-} = \emptyset \quad \mathtt{ord}\ \mathit{vars}\ \mathtt{in}\ P = \overrightarrow{\alpha}}{\mathtt{ord}\ \mathit{vars}\ \mathtt{in} \exists \overrightarrow{\alpha^-}.\ P = \overrightarrow{\alpha}}\ (\exists^{\mathit{ORD}})$$

Analogously, the variable can be ordered in an algorithmic *type (*$\mathtt{ord}\ \mathit{vars}\ \mathtt{in}\ P$ *and* $\mathtt{ord}\ \mathit{vars}\ \mathtt{in}\ N$*). In these cases, we treat the algorithmic variables as if they were declarative variables.*

Next, we use the variable ordering in the normalization procedure. Specifically, normalization recursively traverses the type, and for each quantified case reorders the quantified variables in a canonical order dictated by Algorithm 4, removing unused ones.

Algorithm 5 (Type Normalization)**.**

$\boxed{\mathtt{nf}\ (N) = M}$ $\boxed{\mathtt{nf}\ (P) = Q}$

$$\frac{}{\mathtt{nf}\ (\alpha^-) = \alpha^-}\ (\mathit{VAR}^{\mathit{NF}}_-) \qquad \frac{}{\mathtt{nf}\ (\alpha^+) = \alpha^+}\ (\mathit{VAR}^{\mathit{NF}}_+)$$

$$\frac{\mathtt{nf}\ (P) = Q}{\mathtt{nf}\ (\uparrow P) = \uparrow Q}\ (\uparrow^{\mathit{NF}}) \qquad \frac{\mathtt{nf}\ (N) = M}{\mathtt{nf}\ (\downarrow N) = \downarrow M}\ (\downarrow^{\mathit{NF}})$$

$$\frac{\mathtt{nf}\ (P) = Q \quad \mathtt{nf}\ (N) = M}{\mathtt{nf}\ (P \to N) = Q \to M}\ (\to^{\mathit{NF}}) \qquad \frac{\mathtt{nf}\ (P) = P' \quad \mathtt{ord}\ \overrightarrow{\alpha^-}\ \mathtt{in}\ P' = \overrightarrow{\alpha^-{}'}}{\mathtt{nf}\ (\exists \overrightarrow{\alpha^-}.\ P) = \exists \overrightarrow{\alpha^-{}'}.\ P'}\ (\exists^{\mathit{NF}})$$

$$\frac{\mathtt{nf}\ (N) = N' \quad \mathtt{ord}\ \overrightarrow{\alpha^+}\ \mathtt{in}\ N' = \overrightarrow{\alpha^+{}'}}{\mathtt{nf}\ (\forall \overrightarrow{\alpha^+}.\ N) = \forall \overrightarrow{\alpha^+{}'}.\ N'}\ (\forall^{\mathit{NF}})$$

Analogously, we define the normalization of algorithmic types by adding base cases:

$\boxed{\mathtt{nf}\ (N) = M}$ $\boxed{\mathtt{nf}\ (P) = Q}$

$$\frac{}{\mathtt{nf}\ (\widehat{\alpha^-}) = \widehat{\alpha^-}}\ (\mathit{UVAR}^{\mathit{NF}}_-) \qquad \frac{}{\mathtt{nf}\ (\widehat{\alpha^+}) = \widehat{\alpha^+}}\ (\mathit{UVAR}^{\mathit{NF}}_+)$$

Lemma 48 demonstrates that the equivalence of types is the same as the equality of their normal forms.

Theorem (Correctness of Normalization)**.** *Assuming the types are well-formed in* Θ*,*

- $\Theta \vdash N \simeq^{\leqslant} M$ *if and only if* $\mathtt{nf}\ (N) = \mathtt{nf}\ (M)$*;*
- $\Theta \vdash P \simeq^{\leqslant} Q$ *if and only if* $\mathtt{nf}\ (P) = \mathtt{nf}\ (Q)$*.*

Algorithm 6 (Substitution Normalization)**.** *For a substitution* σ*, we define* $\mathtt{nf}\ (\sigma)$ *as a substitution that maps* $\alpha^\pm$ *into* $\mathtt{nf}\ ([\sigma]\alpha^\pm)$*.*

The rest of this chapter is devoted to the central algorithm of the type system—the subtyping algorithm.

A.2.7 Subtyping

Now, we present the subtyping algorithm itself. Although the algorithm is presented as a single procedure, is important for the structure of the proof that the positive subtyping algorithm does not invoke the negative one. This way, the correctness of the positive subtyping will be proved independently and used afterwards to prove the correctness of the negative subtyping.

Algorithm 7 (Subtyping)**.**

$\boxed{\Theta;\ \Xi \vDash N \leqslant M \dashv C}$ *Negative subtyping*

$$\frac{}{\Theta;\ \Xi \vDash \alpha^- \leqslant \alpha^- \dashv \cdot}\ (\mathit{VAR}^{\leqslant}_{-})$$

$$\frac{\Theta; \Xi \vDash \mathtt{nf}\,(P) \overset{u}{\simeq} \mathtt{nf}\,(Q) \dashv UC}{\Theta;\ \Xi \vDash \uparrow P \leqslant \uparrow Q \dashv UC}\ (\uparrow^{\leqslant})$$

$$\frac{\begin{array}{c}\overrightarrow{\widehat{\alpha}^+}\ \mathtt{are\ fresh} \\ \Theta, \overrightarrow{\beta^+};\ \Xi, \overrightarrow{\widehat{\alpha}^+}\{\Theta, \overrightarrow{\beta^+}\} \vDash [\overrightarrow{\widehat{\alpha}^+}/\overrightarrow{\alpha^+}]N \leqslant M \dashv C\end{array}}{\Theta;\ \Xi \vDash \forall \overrightarrow{\alpha^+}.\ N \leqslant \forall \overrightarrow{\beta^+}.\ M \dashv C \setminus \overrightarrow{\widehat{\alpha}^+}}\ (\forall^{\leqslant})$$

$$\frac{\begin{array}{c}\Theta;\ \Xi \vDash P \geqslant Q \dashv C_1 \\ \Theta;\ \Xi \vDash N \leqslant M \dashv C_2 \\ \Xi \vdash C_1\ \&\ C_2 = C\end{array}}{\Theta;\ \Xi \vDash P \to N \leqslant Q \to M \dashv C}\ (\to^{\leqslant})$$

$\boxed{\Theta;\ \Xi \vDash P \geqslant Q \dashv C}$ *Positive supertyping*

$$\frac{}{\Theta;\ \Xi \vDash \alpha^+ \geqslant \alpha^+ \dashv \cdot}\ (\mathit{VAR}^{\geqslant}_{+})$$

$$\frac{\begin{array}{c}\overrightarrow{\widehat{\alpha}^-}\ \mathtt{are\ fresh} \\ \Theta, \overrightarrow{\beta^-};\ \Xi, \overrightarrow{\widehat{\alpha}^-}\{\Theta, \overrightarrow{\beta^-}\} \vDash [\overrightarrow{\widehat{\alpha}^-}/\overrightarrow{\alpha^-}]P \geqslant Q \dashv C\end{array}}{\Theta;\ \Xi \vDash \exists \overrightarrow{\alpha^-}.\ P \geqslant \exists \overrightarrow{\beta^-}.\ Q \dashv C \setminus \overrightarrow{\widehat{\alpha}^-}}\ (\exists^{\geqslant})$$

$$\frac{\Theta; \Xi \vDash \mathtt{nf}\,(N) \overset{u}{\simeq} \mathtt{nf}\,(M) \dashv UC}{\Theta;\ \Xi \vDash \downarrow N \geqslant \downarrow M \dashv UC}\ (\downarrow^{\geqslant})$$

$$\frac{\mathtt{upgrade}\ \Theta \vdash P\ \mathtt{to}\ \Xi(\widehat{\alpha}^+) = Q}{\Theta;\ \Xi \vDash \widehat{\alpha}^+ \geqslant P \dashv (\widehat{\alpha}^+ :\geqslant Q)}\ (\mathit{UVAR}^{\geqslant})$$

The inputs of the subtyping algorithm are the declarative context Θ, the subtyping context Ξ (it specifies in which contexts the algorithmic variables must be instantiated), and the types themselves: N and M for the negative case, and P and Q for the positive case. As one of the invariants, we require M and Q to be declarative (i.e. not containing algorithmic variables). The output of the algorithm is a set of *subtyping constraints* C, which will be discussed in the next section.

Let us overview the inference rules of the subtyping algorithm.

- $(\textsc{Var}^{\leqslant}_{-})$ and $(\textsc{Var}^{\geqslant}_{+})$ are the base cases. They copy the corresponding declarative rules and ensure reflexivity.
- $(\textsc{UVar}^{\geqslant})$ is the only case generating subtyping constraints. In this case, we must ensure that the resulting constraints guarantee that the instantiation of $\widehat{\alpha}^{+}$ is a supertype of P. However, the obvious constraint $\widehat{\alpha}^{+} :\geqslant P$ might be problematic if P is not well-formed in $\Xi(\widehat{\alpha}^{+})$. For this reason, we use the *upgrade* procedure (it will be covered in Section A.2.10) to find the minimal supertype of P, which is well-formed in $\Xi(\widehat{\alpha}^{+})$.

  Notice that this rule does not have a negative counterpart. This is because one of the important invariants of the algorithm: in the negative subtyping, only positive algorithmic variables can occur in the types.
- $(\downarrow^{\geqslant})$ and $(\uparrow^{\leqslant})$ are the *shift* rules. According to the declarative system, shifted subtyping requires equivalence. In the presence of the algorithmic variables, it means that the left and the right-hand sides of the subtyping must be unified. Hence, the shift rules invoke the unification algorithm, which will be discussed in Section A.2.9. The unification returns the minimal set of constraints UC, which is necessary and sufficient for the subtyping.
- $(\rightarrow^{\leqslant})$. In this case, the algorithm makes two calls: a recursive call to the negative subtyping algorithm for the argument types, and a call to the positive subtyping algorithm for the result types. After that, the resulting constraints are merged using the *subtyping constraint merge* procedure, which is discussed in Section A.2.8.
- $(\forall^{\leqslant})$ and $(\exists^{\geqslant})$ are symmetric. These are the only places where the algorithmic variables are introduced. It is done by algorithmization (Section A.2.3) of the quantified variables: these variables are replaced by fresh algorithmic variables in the body of the quantified type, the algorithmic variables are added to the subtyping context Ξ, after that, the recursive call is made. Notice that the declarative context Θ is extended by the quantified variables from the right-hand side, which matches the declarative system.

Then soundness lemma (Lemmas 86 and 92) and completeness (Lemmas 87 and 93) of the algorithm together give us the following simplified theorem:

Theorem (Correctness of subtyping algorithm)**.**

– $\Theta;\ \cdot \vDash N \leqslant M \dashv \cdot$ *is equivalent to* $\Theta \vdash N \leqslant M$*;*

\+ $\Theta;\ \cdot \vDash P \geqslant Q \dashv \cdot$ *is equivalent to* $\Theta \vdash P \geqslant Q$.

A.2.8 Constraints

Unification and subtyping algorithms are based on constraint generation. The constraints are represented by a set of constraint entries.

Definition 19 (Unification Constraint)**.**

unification entry *(denoted as ue) is an expression of shape* $\widehat{\alpha}^{+} :\simeq P$ *or* $\widehat{\alpha}^{-} :\simeq N$*;*

unification constraint *(denoted as UC) is a set of unification constraint entries. We denote* $\{\widehat{\alpha}^{\pm} \mid ue \in UC \text{ restricting } \widehat{\alpha}^{\pm}\}$ *as* $\mathtt{dom}(UC)$.

However, in the subtyping, we need to consider more general kind of constraints. Specifically, subtyping constraint entries can restrict a variable not only to be equivalent to a certain type, but also to be a supertype of a positive type.

Definition 20 (Subtyping Constraint)**.**

subtyping entry *(denoted as e) is an expression of shape* $\widehat{\alpha}^{+} :\geqslant P$, $\widehat{\alpha}^{-} :\simeq N$, *or* $\widehat{\alpha}^{+} :\simeq P$;

subtyping constraint *(denoted as C) is a set of subtyping constraint entries. We denote* $\{\widehat{\alpha}^{\pm} \mid e \in C \text{ restricting } \widehat{\alpha}^{\pm}\}$ *as* $\mathtt{dom}(C)$.

Definition 21 (Well-formed Constraint Entry)**.** *We say that a constraint entry is well-formed in a context* Θ *if its associated type is well-formed in* Θ.

$\Theta \vdash \widehat{\alpha}^{+} :\geqslant P$ *iff* $\Theta \vdash P$;

$\Theta \vdash \widehat{\alpha}^{+} :\simeq P$ *iff* $\Theta \vdash P$;

$\Theta \vdash \widehat{\alpha}^{-} :\simeq N$ *iff* $\Theta \vdash N$.

Definition 22 (Well-formed Constraint)**.** *We say that a constraint is well-formed in a subtyping context* Ξ *if all its entries are well-formed in the corresponding elements of* Ξ. *More formally,* $\Xi \vdash C$ *holds iff for every* $e \in C$, *such that e restricts* $\widehat{\alpha}^{\pm}$, *we have* $\Xi(\widehat{\alpha}^{\pm}) \vdash e$.

We write $\Xi \vdash C : \widehat{\Theta}$ *to denote that* $\Xi \vdash C$ *and* $\mathtt{dom}(C) = \widehat{\Theta}$.

$\Xi \vdash UC$ *and* $\Xi \vdash UC : \widehat{\Theta}$ *are defined analogously.*

A.2.8.1 Constraint Satisfaction. A constraint entry restricts a type that can be assigned to a variable. We say that a type satisfies a constraint entry if it can be assigned to the variable restricted by the entry.

Definition 23 (Type Satisfying a Constraint Entry)**.**

$\boxed{\Theta \vdash N : e}$ *Negative entry satisfaction*

$$\frac{\Theta \vdash N \simeq^{\leqslant} M}{\Theta \vdash N : (\widehat{\alpha}^{-} :\simeq M)}\ (:\simeq^{SAT}_{-})$$

$\boxed{\Theta \vdash P : e}$ *Positive entry satisfaction*

$$\frac{\Theta \vdash P \geqslant Q}{\Theta \vdash P : (\widehat{\alpha}^{+} :\geqslant Q)}\ (:\geqslant^{SAT}_{+})$$

$$\frac{\Theta \vdash P \simeq^{\leqslant} Q}{\Theta \vdash P : (\widehat{\alpha}^{+} :\simeq Q)}\ (:\simeq^{SAT}_{+})$$

We say that a substitution satisfies a constraint—a set of constraint entries if each entry is satisfied by the type assigned to the variable by the substitution.

Definition 24 (Substitution Satisfying a Constraint)**.** *We write* $\Xi \vdash \widehat{\sigma} : C$ *to denote that a substitution* $\widehat{\sigma}$ *satisfies a constraint C in a context* Ξ. *It presumes that* $\Xi \vdash C$ *and means that for any* $ue \in C$, *if ue restricts* $\widehat{\alpha}^{\pm}$, *then* $\Xi(\widehat{\alpha}^{\pm}) \vdash [\widehat{\sigma}]\widehat{\alpha}^{\pm} : ue$.

Unification constraint satisfaction $\Xi \vdash \widehat{\sigma} : UC$ *is defined analogously as a special case of subtyping constraint satisfaction.*

Notice that $\Xi \vdash \widehat{\sigma} : C$ does not imply the signature $\Xi \vdash \widehat{\sigma} : \mathtt{dom}(C)$, because the latter also specifies $\widehat{\sigma}$ outside of the domain $\mathtt{dom}(C)$ (see Definition 16).

A.2.8.2 Constraint Merge. In this section, define the least upper bound for constraints, which we call *merge*. Intuitively, the merge of two constraints is the least constraint such that any substitution satisfying both constraints satisfies the merge as well. First, we define the merge of entries, and then extend it to the set of entries.

Definition 25 (Matching Entries)**.** *We call two unification constraint entries or two subtyping constraint entries matching if they are restricting the same unification variable.*

Two matching entries formed in the same context Θ can be merged in the following way:

Algorithm 8 (Merge of Matching Constraint Entries)**.**

$\boxed{\Theta \vdash e_1 \,\&\, e_2 = e_3}$ *Subtyping Constraint Entry Merge*

$$\frac{\Theta \vDash P_1 \vee P_2 = Q}{\Theta \vdash (\widehat{\alpha}^+ :\geqslant P_1) \,\&\, (\widehat{\alpha}^+ :\geqslant P_2) = (\widehat{\alpha}^+ :\geqslant Q)} \; (\geqslant \&^+ \geqslant)$$

$$\frac{\Theta;\, \cdot \vDash P \geqslant Q \dashv \cdot}{\Theta \vdash (\widehat{\alpha}^+ :\simeq P) \,\&\, (\widehat{\alpha}^+ :\geqslant Q) = (\widehat{\alpha}^+ :\simeq P)} \; (\simeq \&^+ \geqslant)$$

$$\frac{\Theta;\, \cdot \vDash Q \geqslant P \dashv \cdot}{\Theta \vdash (\widehat{\alpha}^+ :\geqslant P) \,\&\, (\widehat{\alpha}^+ :\simeq Q) = (\widehat{\alpha}^+ :\simeq Q)} \; (\geqslant \&^+ \simeq)$$

$$\frac{\mathtt{nf}\,(P) = \mathtt{nf}\,(P')}{\Theta \vdash (\widehat{\alpha}^+ :\simeq P) \,\&\, (\widehat{\alpha}^+ :\simeq P') = (\widehat{\alpha}^+ :\simeq P)} \; (\simeq \&^+ \simeq)$$

$$\frac{\mathtt{nf}\,(N) = \mathtt{nf}\,(N')}{\Theta \vdash (\widehat{\alpha}^- :\simeq N) \,\&\, (\widehat{\alpha}^- :\simeq N') = (\widehat{\alpha}^- :\simeq N)} \; (\simeq \&^- \simeq)$$

- $(\simeq \&^+ \simeq)$ and $(\simeq \&^- \simeq)$ are symmetric cases. To merge two matching entries restricting a variable to be equivalent to certain types, we check that these types are equivalent to each other. To do so, it suffices to check for *equality* of their normal forms, as discussed in Section A.2.6. After that, we return the left-hand entry.
- $(\simeq \&^+ \geqslant)$ and $(\geqslant \&^+ \simeq)$ are also symmetric. In this case, since one of the entries requires the variable to be equal to a type, the resulting entry must also imply that. However, for the soundness, it is needed to ensure that the equating restriction is stronger than the subtyping restriction. For this purpose, the premise invokes the positive subtyping.
- $(\geqslant \&^+ \geqslant)$ In this case, we find the least upper bound of the types from the input restrictions, and as the output, restrict the variable to be a supertype of the result. The least upper bound procedure will be discussed in Section A.2.10.

Unification constraint entries are a special case of subtyping constraint entries. They are merged using the same algorithm (Algorithm 8). Notice that the merge of two matching unification constraint entries is a unification constraint entry.

Lemma 1 (Merge of Matching Unification Constraint Entries is well-defined)**.** *Suppose that $\Theta \vdash ue_1$ and $\Theta \vdash ue_2$ are unification constraint entries. Then the merge of ue_1 and ue_2 $\Theta \vdash ue_1 \,\&\, ue_2 = ue$ according to Algorithm 8, is a unification constraint entry.*

Proof Since ue_1 and ue_2 are matching unification constraint entries, they have the shape $(\widehat{\alpha}^+ :\simeq P_1, \widehat{\alpha}^+ :\simeq P_2)$ or $(\widehat{\alpha}^- :\simeq N_1, \widehat{\alpha}^- :\simeq N_2)$. Then the merge of ue_1 and ue_2 can only be defined by $(\simeq \&^+ \simeq)$ or $(\simeq \&^- \simeq)$. In both cases the result, if it exists, is a unification constraint entry: in the first case, the result has shape $\widehat{\alpha}^+ :\simeq P_1$, in the second case, the result has shape $\widehat{\alpha}^- :\simeq N_1$. ■

Algorithm 9 (Merge of Subtyping Constraints)**.** *Suppose that* $\Xi \vdash C_1$ *and* $\Xi \vdash C_2$. *Then* $\Xi \vdash C_1 \,\&\, C_2 = C$ *defines a set of constraints* C *such that* $e \in C$ *iff either:*

- $e \in C_1$ *and there is no matching* $e' \in C_2$*; or*
- $e \in C_2$ *and there is no matching* $e' \in C_1$*; or*
- $\Xi(\widehat{\alpha}^{\pm}) \vdash e_1 \,\&\, e_2 = e$ *for some* $e_1 \in C_1$ *and* $e_2 \in C_2$ *such that* e_1 *and* e_2 *both restrict variable* $\widehat{\alpha}^{\pm}$.

Unification constraints can be considered as a special case of subtyping constraints, and the merge of unification constraints is defined as the merge of subtyping constraints. Then it is easy to see that the merge of two unification constraints is a unification constraint.

Lemma 2 (Merge of Unification Constraints is well-defined)**.** *Suppose that* $\Xi \vdash UC_1$ *and* $\Xi \vdash UC_2$ *are unification constraints. Then the merge of* UC_1 *and* UC_2 $\Xi \vdash UC_1 \,\&\, UC_2 = UC$ *according to Algorithm 9, is a unification constraint.*

Proof UC consists of unmatched entries of UC_1 and UC_2, which are *unification* constraint entries by assumption, and merge of matching entries, which also are *unification* constraint entries by Lemma 1. ■

Lemmas 89 and 91 show the correctness and initiality of the merge operation, which can be expressed in the following simplified theorem:

Theorem (Correctness of Constraint Merge)**.** *A substitution* $\widehat{\sigma}$ *satisfying both constraints* C_1 *and* C_2 *if and only if it satisfies their merge.*

The unification constraint merge satisfies the same theorem, however, because the merge of unification constraint entries ue_1 and ue_2 always results in one of them, a stronger soundness property holds (see Lemma 69):

Theorem (Soundness of Unification Constraint Merge)**.** *If* $\Xi \vdash UC_1 \,\&\, UC_2 = UC$ *then* $UC = UC_1 \cup UC_2$.

A.2.9 Unification

The subtyping algorithm calls the following subtask: given two algorithmic types, we need to find the most general substitution for the algorithmic variables in these types, such that the resulting types are equivalent. This problem is known as *unification*.

In our case, the unification is restricted in the following way: first, before unifying the types, we normalize them, which allows us to reduce (non-trivial) equivalence to (trivial) equality; second, we preserve invariants which guarantee that one side of the unification is always declarative, which in fact, reduces the unification to the *matching* problem.

The unification procedure returns a set of minimal constraints, that must be satisfied by a substitution unifying the input types.

Algorithm 10 (Unification)**.**

$\boxed{\Theta; \Xi \vDash N \overset{u}{\simeq} M \dashv UC}$ *Negative unification* $\qquad$ $\boxed{\Theta; \Xi \vDash P \overset{u}{\simeq} Q \dashv UC}$ *Positive unification*

$$\frac{}{\Theta; \Xi \vDash \alpha^- \overset{u}{\simeq} \alpha^- \dashv \cdot}\ (\textsc{Var}^{\overset{u}{\simeq}}_-) \qquad \frac{}{\Theta; \Xi \vDash \alpha^+ \overset{u}{\simeq} \alpha^+ \dashv \cdot}\ (\textsc{Var}^{\overset{u}{\simeq}}_+)$$

$$\frac{\Theta; \Xi \vDash P \overset{u}{\simeq} Q \dashv UC}{\Theta; \Xi \vDash \uparrow P \overset{u}{\simeq} \uparrow Q \dashv UC}\ (\uparrow^{\overset{u}{\simeq}}) \qquad \frac{\Theta; \Xi \vDash N \overset{u}{\simeq} M \dashv UC}{\Theta; \Xi \vDash \downarrow N \overset{u}{\simeq} \downarrow M \dashv UC}\ (\downarrow^{\overset{u}{\simeq}})$$

$$\frac{\Theta; \Xi \vDash P \overset{u}{\simeq} Q \dashv UC_1 \quad \Theta; \Xi \vDash N \overset{u}{\simeq} M \dashv UC_2}{\Theta; \Xi \vDash P \to N \overset{u}{\simeq} Q \to M \dashv UC_1 \,\&\, UC_2}\ (\to^{\overset{u}{\simeq}}) \qquad \frac{\Theta, \overrightarrow{\alpha^+}; \Xi \vDash P \overset{u}{\simeq} Q \dashv UC}{\Theta; \Xi \vDash \exists \overrightarrow{\alpha^+}.\ P \overset{u}{\simeq} \exists \overrightarrow{\alpha^+}.\ Q \dashv UC}\ (\exists^{\overset{u}{\simeq}})$$

$$\frac{\Theta, \overrightarrow{\alpha^+}; \Xi \vDash N \overset{u}{\simeq} M \dashv UC}{\Theta; \Xi \vDash \forall \overrightarrow{\alpha^+}.\ N \overset{u}{\simeq} \forall \overrightarrow{\alpha^+}.\ M \dashv UC}\ (\forall^{\overset{u}{\simeq}}) \qquad \frac{\Xi(\widehat{\alpha}^+) \vdash P}{\Theta; \Xi \vDash \widehat{\alpha}^+ \overset{u}{\simeq} P \dashv (\widehat{\alpha}^+ :\simeq P)}\ (\textsc{UVar}^{\overset{u}{\simeq}}_+)$$

$$\frac{\Xi(\widehat{\alpha}^-) \vdash N}{\Theta; \Xi \vDash \widehat{\alpha}^- \overset{u}{\simeq} N \dashv (\widehat{\alpha}^- :\simeq N)}\ (\textsc{UVar}^{\overset{u}{\simeq}}_-)$$

- $(\uparrow^{\overset{u}{\simeq}})$, $(\downarrow^{\overset{u}{\simeq}})$, $(\forall^{\overset{u}{\simeq}})$, and $(\exists^{\overset{u}{\simeq}})$ are defined congruently. In the shift rules, the algorithm removes the outermost constructor. In the $\forall$ and $\exists$ rules, it removes the quantifiers, adding the quantified variables to the context Θ. Notice that Ξ, which specifies the contexts in which the algorithmic variables must be instantiated, is not changed.
- $(\textsc{Var}^{\overset{u}{\simeq}}_-)$ and $(\textsc{Var}^{\overset{u}{\simeq}}_+)$ are the base cases. Since the sides are equal and free from algorithmic variables, the unification returns an empty constraint.
- $(\textsc{Var}^{\overset{u}{\simeq}}_-)$ and $(\textsc{Var}^{\overset{u}{\simeq}}_+)$ are symmetric cases constructing the constraints. When an algorithmic variable is unified with a type, we must check that the type is well-formed in the required context, and if it is, we return a constraint restricting the variable to be equivalent to that type.
- $(\to^{\overset{u}{\simeq}})$. In this case, the algorithm makes two recursive calls: it unifies the arguments and the results of the arrows. After that, the resulting constraints are merged using the *unification constraint merge* procedure, which is discussed in Section A.2.8. Notice that UC_1 and UC_2 are guaranteed to be *unification* constraints, not arbitrary *subtyping* constraints: it is important for modularizing the proofs, since the properties of the *unification* constraint merge can be proved independently from the *subtyping* constraint merge.

A.2.10 Least Upper Bound

In this section, we present the algorithm finding the least common supertype of two positive types. It is used directly by the constraint merge procedure (Section A.2.8), and indirectly, through the type upgrade by positive subtyping (Section A.2.7). Perhaps, the least upper

bound is the least intuitive part of the algorithm, and its correctness will be covered in Section C.3.8.

Algorithm 11 (The Least Upper Bound Algorithm)**.**

$\boxed{\Theta \vDash P_1 \vee P_2 = Q}$ *Least Upper Bound*

$$\frac{\Theta, \overrightarrow{\alpha^-}, \overrightarrow{\beta^-} \vDash P_1 \vee P_2 = Q}{\Theta \vDash \exists \overrightarrow{\alpha^-}.\, P_1 \vee \exists \overrightarrow{\beta^-}.\, P_2 = Q}\ (\exists^\vee)$$

$$\frac{}{\Theta \vDash \alpha^+ \vee \alpha^+ = \alpha^+}\ (\textsc{Var}^\vee)$$

$$\frac{\Theta \vDash \mathtt{nf}\,(\downarrow N) \overset{a}{\cong} \mathtt{nf}\,(\downarrow M) \dashv (\widehat{\Theta}, P, \widehat{\tau_1}, \widehat{\tau_2})}{\Theta \vDash \downarrow N \vee \downarrow M = \exists \overrightarrow{\alpha^-}.\, [\overrightarrow{\alpha^-}/\widehat{\Theta}]P}\ (\downarrow^\vee)$$

- ($\textsc{Var}^\vee$) The base case is trivial: the least upper bound of to equal variables is the variable itself.
- ($\downarrow^\vee$) In case both sides of the least upper bound are shifted, the algorithm needs to find the anti-unifier of them. Intuitively, this is because in general, the upper bounds of $\downarrow N$ are $\exists \overrightarrow{\alpha^-}.\, P$ such that $\overrightarrow{\alpha^-}$ can be instantiated with some $\overrightarrow{M}$ so that $\Theta \vdash [\overrightarrow{M}/\overrightarrow{\alpha^-}]P \simeq^{\leqslant} \downarrow N$ (see Lemma 77).
- ($\exists^\vee$) In this case, we move the quantified variables to the context Θ, and make a recursive call. It is important to make sure that $\overrightarrow{\alpha^-}$ and $\overrightarrow{\beta^-}$ are disjoint. In this case, it is guaranteed that the resulting $\mathtt{fv}\,(Q)$ will be free of $\overrightarrow{\alpha^-}$ and $\overrightarrow{\beta^-}$, and thus, the resulting type will be a supertype of both sides (it will be discussed in Lemma 77).

In the positive subtyping algorithm (Section A.2.7), ($\textsc{UVar}^{\geqslant}$) generates a restriction of a variable $\widehat{\alpha}^+$. On the one hand, this restriction must imply $\widehat{\alpha}^+ :\geqslant P$ for the subtyping to hold. On the other hand, the type used in this restriction must be well-formed in a potentially stronger (smaller) context than P.

To resolve this problem, we define the *upgrade* procedure, which for given Θ_0, $\overrightarrow{\alpha^\pm}$, and $\Theta_0, \overrightarrow{\alpha^\pm} \vdash P$, finds $\Theta_0 \vdash Q$—the least supertype of P among the types well-formed in Θ_0.

The trick is to make sure that the 'forbidden' variables $\overrightarrow{\alpha^\pm}$ are not used explicitly in the supertypes of P. For this purpose, we construct new types P_1 and P_2, in each of them replacing the forbidden variables with fresh variables $\overrightarrow{\beta^\pm}$ and $\overrightarrow{\gamma^\pm}$, and then find the least upper bound of P_1 and P_2. It turns out that this renaming forces the common types of P_1 and P_2 to be agnostic to $\overrightarrow{\alpha^\pm}$, and thus, the supertypes of P well-formed in Θ_0 are exactly the common supertypes of P_1 and P_2. These properties are considered in more details in Section C.3.9.

Algorithm 12 (Type Upgrade)**.**

$\boxed{\mathtt{upgrade}\ \Theta \vdash P\ \mathtt{to}\ \Theta_0 = Q}$ *Type Upgrade*

$$\frac{\begin{array}{l}\Theta = \Theta_0, \overrightarrow{\alpha^\pm} \\ \overrightarrow{\beta^\pm}\ \mathtt{are\ fresh} \quad \overrightarrow{\gamma^\pm}\ \mathtt{are\ fresh} \\ \Theta_0, \overrightarrow{\beta^\pm}, \overrightarrow{\gamma^\pm} \vDash [\overrightarrow{\beta^\pm}/\overrightarrow{\alpha^\pm}]P \vee [\overrightarrow{\gamma^\pm}/\overrightarrow{\alpha^\pm}]P = Q\end{array}}{\mathtt{upgrade}\ \Theta \vdash P\ \mathtt{to}\ \Theta_0 = Q}\ (\textsc{Up}_G)$$

A.2.10.1 Note on the Greatest Lower Bound. In contrast to the least upper bound, the general greatest lower bound does not exist in our system. For instance, consider a positive type P, together with its non-equivalent supertypes P_1 and $P_2 \not\simeq P_1$ (for example, $P = \downarrow\uparrow\downarrow\gamma^-$, $P_1 = \exists\alpha^-.\downarrow\uparrow\downarrow\alpha^-$, and $P_2 = \exists\alpha^-.\downarrow\alpha^-$). Then for arbitrary Q and N, let us consider the common subtypes of $A = Q \to \downarrow\uparrow Q \to \downarrow\uparrow Q \to N$ and $B = P \to \downarrow\uparrow P_1 \to \downarrow\uparrow P_2 \to N$. It is easy to see that $\forall\alpha^+.\forall\beta^+.\alpha^+ \to \downarrow\uparrow\alpha^+ \to \downarrow\uparrow\beta^+ \to N$ and $\forall\alpha^+.\forall\beta^+.\alpha^+ \to \downarrow\uparrow\beta^+ \to \downarrow\uparrow\alpha^+ \to N$ are both *maximal* common subtypes of A and B, and since they are not equivalent, none of them is the *greatest* one.

However, we designed the subtyping system in such a way that the greatest lower bound is not needed: the negative variables are always 'protected' by *invariant* shifts ($\uparrow$ and $\downarrow$), and thus, the algorithm can only require a substitution of a negative variable to be *equivalent* to some type but never to be a *subtype*.

A.2.11 Anti-unification

Next, we define the anti-unification procedure, also known as the *most specific generalization*. As an input, it takes two declarative types (e.g., in the positive case P_1 and P_2) and a context Θ. and returns a type Q—the generalizer, containing negative placeholders (represented by algorithmic variables) from $\widehat{\Theta}$ and two substitutions $\widehat{\tau}_1$ and $\widehat{\tau}_2$. The substitutions replace the placeholders with declarative types well-formed in Θ, such that $[\widehat{\tau}_1]Q = P_1$ and $[\widehat{\tau}_2]Q = P_2$. Moreover, the algorithm guarantees that Q is the most specific type with this property: any other generalizer can be turned into Q by some substitution $\widehat{\rho}$.

It is important to note the differences between the standard anti-unification and our version. First, we only allow the placeholders at *negative* positions, which means, for example, that α^+ and β^+ cannot be generalized. Second, the generated pair of substitutions $\widehat{\tau}_1$ and $\widehat{\tau}_2$ must replace the placeholders with types well-formed in a specified context Θ.

The anti-unification algorithm assumes that the input types are normalized. This way, anti-unification up-to-equality rather than anti-unification up-to-equivalence is sufficient.

Algorithm 13 (Anti-unification)**.**

$\boxed{\Theta \vDash P_1 \overset{a}{\simeq} P_2 \dashv (\widehat{\Theta}, Q, \widehat{\tau}_1, \widehat{\tau}_2)}$

$$\frac{}{\Theta \vDash \alpha^+ \overset{a}{\simeq} \alpha^+ \dashv (\cdot, \alpha^+, \cdot, \cdot)}\ (\mathit{VAR}^{\overset{a}{\simeq}}_+)$$

$$\frac{\Theta \vDash N_1 \overset{a}{\simeq} N_2 \dashv (\widehat{\Theta}, M, \widehat{\tau}_1, \widehat{\tau}_2)}{\Theta \vDash \downarrow N_1 \overset{a}{\simeq} \downarrow N_2 \dashv (\widehat{\Theta}, \downarrow M, \widehat{\tau}_1, \widehat{\tau}_2)}\ (\downarrow^{\overset{a}{\simeq}})$$

$$\frac{\overrightarrow{\alpha^-} \cap \Theta = \emptyset \quad \Theta \vDash P_1 \overset{a}{\simeq} P_2 \dashv (\widehat{\Theta}, Q, \widehat{\tau}_1, \widehat{\tau}_2)}{\Theta \vDash \exists\overrightarrow{\alpha^-}.P_1 \overset{a}{\simeq} \exists\overrightarrow{\alpha^-}.P_2 \dashv (\widehat{\Theta}, \exists\overrightarrow{\alpha^-}.Q, \widehat{\tau}_1, \widehat{\tau}_2)}\ (\exists^{\overset{a}{\simeq}})$$

$\boxed{\Theta \vDash N_1 \overset{a}{\simeq} N_2 \dashv (\widehat{\Theta}, M, \widehat{\tau}_1, \widehat{\tau}_2)}$

$$\frac{}{\Theta \vDash \alpha^- \overset{a}{\simeq} \alpha^- \dashv (\cdot, \alpha^-, \cdot, \cdot)}\ (\mathit{VAR}^{\overset{a}{\simeq}}_-)$$

$$\frac{\Theta \vDash P_1 \overset{a}{\cong} P_2 \dashv (\widehat{\Theta}, Q, \widehat{\tau}_1, \widehat{\tau}_2)}{\Theta \vDash \uparrow P_1 \overset{a}{\cong} \uparrow P_2 \dashv (\widehat{\Theta}, \uparrow Q, \widehat{\tau}_1, \widehat{\tau}_2)}\ (\uparrow^{\overset{a}{\cong}})$$

$$\frac{\overrightarrow{\alpha^{+}} \cap \Theta = \emptyset \quad \Theta \vDash N_1 \overset{a}{\cong} N_2 \dashv (\widehat{\Theta}, M, \widehat{\tau}_1, \widehat{\tau}_2)}{\Theta \vDash \forall \overrightarrow{\alpha^{+}}.\, N_1 \overset{a}{\cong} \forall \overrightarrow{\alpha^{+}}.\, N_2 \dashv (\widehat{\Theta}, \forall \overrightarrow{\alpha^{+}}.\, M, \widehat{\tau}_1, \widehat{\tau}_2)}\ (\forall^{\overset{a}{\cong}})$$

$$\frac{\Theta \vDash P_1 \overset{a}{\cong} P_2 \dashv (\widehat{\Theta}_1, Q, \widehat{\tau}_1, \widehat{\tau}_2) \quad \Theta \vDash N_1 \overset{a}{\cong} N_2 \dashv (\widehat{\Theta}_2, M, \widehat{\tau}_1', \widehat{\tau}_2')}{\Theta \vDash P_1 \to N_1 \overset{a}{\cong} P_2 \to N_2 \dashv (\widehat{\Theta}_1 \cup \widehat{\Theta}_2, Q \to M, \widehat{\tau}_1 \cup \widehat{\tau}_1', \widehat{\tau}_2 \cup \widehat{\tau}_2')}\ (\to^{\overset{a}{\cong}})$$

$$\frac{\textit{if other rules are not applicable} \quad \Theta \vdash N \quad \Theta \vdash M}{\Theta \vDash N \overset{a}{\cong} M \dashv (\widehat{\alpha}^{-}_{\{N,M\}}, \widehat{\alpha}^{-}_{\{N,M\}}, (\widehat{\alpha}^{-}_{\{N,M\}} \mapsto N), (\widehat{\alpha}^{-}_{\{N,M\}} \mapsto M))}\ (AU)$$

- ($\textsc{Var}^{\overset{a}{\cong}}_{+}$) and ($\textsc{Var}^{\overset{a}{\cong}}_{-}$) are the base cases. In this case, since the input types are equal, the algorithm returns this type as a generalizer, without generating any placeholders.
- ($\downarrow^{\overset{a}{\cong}}$), ($\uparrow^{\overset{a}{\cong}}$), ($\forall^{\overset{a}{\cong}}$), and ($\exists^{\overset{a}{\cong}}$) are defined congruently. In the shift rules, the algorithm removes the outermost constructor. In the $\forall$ and $\exists$ rules, it removes the quantifiers. Notice that the algorithm does not add the removed variables to the context Θ. This is because Θ is used to restrict the resulting anti-unification substitutions, and is fixed throughout the algorithm.
- (AU) is the most important rule, since it generates the placeholders. This rule only applies if other negative rules failed. Because of that, the anti-unification procedure is *not* syntax-directed.

  The generated placeholder is indexed with a pair of types it is mapped to. It allows the algorithm to automatically unite the anti-unification solutions generated by the different branches of ($\to^{\overset{a}{\cong}}$).

  Notice that this rule does not have a positive counterpart, since we only allow negative placeholders.
- ($\to^{\overset{a}{\cong}}$) makes two recursive calls to the anti-unification procedure, and unites the results. Suppose that $\widehat{\tau}_1$ and $\widehat{\tau}_2$ are the substitutions generated by anti-unification of *argument* types of the arrow, and $\widehat{\tau}_1'$ and $\widehat{\tau}_2'$ are the substitutions generated by anti-unification of *result* types of the arrow. It is important that if $(\widehat{\tau}_1, \widehat{\tau}_2)$ and $(\widehat{\tau}_1', \widehat{\tau}_2')$ send some variables to the same pair of types, i.e., $[\widehat{\tau}_1]\widehat{\alpha}^{-} = [\widehat{\tau}_1']\widehat{\beta}^{-}$ and $[\widehat{\tau}_2]\widehat{\alpha}^{-} = [\widehat{\tau}_2']\widehat{\beta}^{-}$, then these variables are equal, i.e., $\widehat{\alpha}^{-} = \widehat{\beta}^{-}$. This property is guaranteed by (AU): the name of the placeholder is determined by the pair of types it is mapped to.

A.3 Declarative Typing

In the previous section, we presented the type system together with the subtyping specification and the algorithm. In this section, we describe the language under this type system, together with the type inference specification and algorithm.

A.3.1 Grammar

The syntax of $F^{\pm}_{\exists}$ terms is given by the following grammar:

Definition 26 (Grammar of Terms)**.**

Computation Terms

$$
\begin{array}{lll}
c,\ d & ::= & \\
& \mid & (c : N) \\
& \mid & \lambda x : P.\ c \\
& \mid & \Lambda \alpha^{+}.\ c \\
& \mid & \texttt{return}\ v \\
& \mid & \texttt{let}\ x = v;\ c \\
& \mid & \texttt{let}\ x : P = c;\ c' \\
& \mid & \texttt{let}\ x : P = v(\vec{v});\ c \\
& \mid & \texttt{let}\ x = v(\vec{v});\ c \\
& \mid & \texttt{let}^{\exists}(\vec{\alpha}, x) = v;\ c
\end{array}
$$

Value Terms

$$
\begin{array}{lll}
v,\ w & ::= & \\
& \mid & x \\
& \mid & \{c\} \\
& \mid & (v : P)
\end{array}
$$

Notice that the language does not have first-class applications: instead, we use applicative let bindings— constructions that bind a result of a fully applied function to a (positive) variable. In the call-by-push-value paradigm, it corresponds to monadic bind or do-notation. Typewise, these let-binders come in two forms: annotated and unannotated. The annotated let-binders $\texttt{let}\ x : P = v(\vec{v});\ c$ requires the application to infer the annotated P, whereas the unannotated $\texttt{let}\ x = v(\vec{v});\ c$ is used when the inferred type is unique.

A computation of a polymorphic type is constructed using $\Lambda \alpha^{+}.\ c$, however, the elimination of $\forall$ is implicit. Conversely, the existential types are constructed implicitly and eliminated using the standard unpack mechanism: $\texttt{let}^{\exists}(\vec{\alpha}, x) = v;\ c$.

Another dual pair of constructions are $\texttt{return}\ v$ and $\{c\}$. The former allows us to embed a value in pure computations. The latter, on the contrary, encapsulates a thunk of computation in a value.

Finally, the language has several standard constructions: lambda-abstractions $\lambda x : P.\ c$, standard let-bindings $\texttt{let}\ x = v;\ c$, and type annotations that can be added to any value or computation: $(v : P)$ and $(c : N)$.

A.3.2 Declarative Type Inference

Next, we define the specification of the type inference for our language. First, we introduce variable context specifying the types of variables in the scope of the current rule.

Definition 27 (Variable Context)**.** *The variable typing context* Γ *is represented by a set of entries of the form* $x : P$.

The specification is represented by an inference system of three mutually recursive judgments: positive inference $\Theta; \Gamma \vdash v : P$, negative type inference $\Theta; \Gamma \vdash c : N$, and application type inference $\Theta ; \Gamma \vdash N \bullet \vec{v} \Rrightarrow M$. In the premises, the inference rules also refer to the declarative subtyping (Definition 10), type well-formedness (Algorithm 1), and normalization (Algorithm 5).

- $\Theta; \Gamma \vdash v : P$ (and symmetrically, $\Theta; \Gamma \vdash c : N$) means that under the type context Θ and the variable context Γ, for the value v, type P is inferrable. It guarantees that v is well-formed in Θ and Γ in the standard sense.
- $\Theta ; \Gamma \vdash N \bullet \vec{v} \Rrightarrow M$ is the application type inference judgment. It means that if a head of type N is applied to list of values $\vec{v}$, then the resulting computation can be typed as M.

Definition 28 (Declarative Type Inference)**.**

$\boxed{\Theta; \Gamma \vdash c : N}$ *Negative typing*

$$\frac{\Theta \vdash P \quad \Theta; \Gamma, x : P \vdash c : N}{\Theta; \Gamma \vdash \lambda x : P.\, c : P \to N}\ (\lambda^{INF}) \qquad \frac{\Theta; \Gamma \vdash v : \downarrow M \quad \Theta ; \Gamma \vdash M \bullet \vec{v} \Rrightarrow \uparrow Q \ \textit{principal} \quad \Theta; \Gamma, x : Q \vdash c : N}{\Theta; \Gamma \vdash \mathtt{let}\, x = v(\vec{v});\ c : N}\ (LET^{INF}_{@})$$

$$\frac{\Theta, \alpha^{+}; \Gamma \vdash c : N}{\Theta; \Gamma \vdash \Lambda \alpha^{+}.\, c : \forall \alpha^{+}.\, N}\ (\Lambda^{INF}) \qquad \frac{\Theta \vdash P \quad \Theta; \Gamma \vdash v : \downarrow M \quad \Theta ; \Gamma \vdash M \bullet \vec{v} \Rrightarrow M' \quad \Theta \vdash M' \leqslant \uparrow P \quad \Theta; \Gamma, x : P \vdash c : N}{\Theta; \Gamma \vdash \mathtt{let}\, x : P = v(\vec{v});\ c : N}\ (LET^{INF}_{:@})$$

$$\frac{\Theta; \Gamma \vdash v : P}{\Theta; \Gamma \vdash \mathtt{return}\, v : \uparrow P}\ (RET^{INF}) \qquad \frac{\Theta; \Gamma \vdash v : \exists \overrightarrow{\alpha^{-}}.\, P \quad \mathtt{nf}\,(\exists \overrightarrow{\alpha^{-}}.\, P) = \exists \overrightarrow{\alpha^{-}}.\, P \quad \Theta, \overrightarrow{\alpha^{-}}; \Gamma, x : P \vdash c : N \quad \Theta \vdash N}{\Theta; \Gamma \vdash \mathtt{let}^{\exists}(\overrightarrow{\alpha^{-}}, x) = v;\ c : N}\ (LET^{INF}_{\exists})$$

$$\frac{\Theta; \Gamma \vdash v : P \quad \Theta; \Gamma, x : P \vdash c : N}{\Theta; \Gamma \vdash \mathtt{let}\, x = v;\ c : N}\ (LET^{INF}) \qquad \frac{\Theta \vdash M \quad \Theta; \Gamma \vdash c : N \quad \Theta \vdash N \leqslant M}{\Theta; \Gamma \vdash (c : M) : M}\ (ANN^{INF}_{-})$$

$$\frac{\Theta \vdash P \quad \Theta; \Gamma \vdash c : M \quad \Theta \vdash M \leqslant \uparrow P \quad \Theta; \Gamma, x : P \vdash c' : N}{\Theta; \Gamma \vdash \mathtt{let}\, x : P = c;\ c' : N}\ (LET^{INF}_{C}) \qquad \frac{\Theta; \Gamma \vdash c : N \quad \Theta \vdash N \simeq^{\leqslant} N'}{\Theta; \Gamma \vdash c : N'}\ (\simeq^{INF}_{-})$$

$\boxed{\Theta; \Gamma \vdash v : P}$ *Positive typing*

$$\frac{x : P \in \Gamma}{\Theta; \Gamma \vdash x : P}\ (VAR^{INF}) \qquad \frac{\Theta \vdash Q \quad \Theta; \Gamma \vdash v : P \quad \Theta \vdash Q \geqslant P}{\Theta; \Gamma \vdash (v : Q) : Q}\ (ANN^{INF}_{+})$$

$$\frac{\Theta; \Gamma \vdash c : N}{\Theta; \Gamma \vdash \{c\} : \downarrow N}\ (\{\}^{INF}) \qquad \frac{\Theta; \Gamma \vdash v : P \quad \Theta \vdash P \simeq^{\leqslant} P'}{\Theta; \Gamma \vdash v : P'}\ (\simeq^{INF}_{+})$$

$\boxed{\Theta ; \Gamma \vdash N \bullet \vec{v} \Rrightarrow M}$ *Application typing*

$$\frac{\Theta \vdash N \simeq^{\leqslant} N'}{\Theta ; \Gamma \vdash N \bullet \cdot \Rrightarrow N'}\ (\emptyset^{INF}_{\bullet\Rrightarrow}) \qquad \frac{\vec{v} \neq \cdot \quad \overrightarrow{\alpha^{+}} \neq \cdot \quad \Theta \vdash \sigma : \overrightarrow{\alpha^{+}} \quad \Theta ; \Gamma \vdash [\sigma]N \bullet \vec{v} \Rrightarrow M}{\Theta ; \Gamma \vdash \forall \overrightarrow{\alpha^{+}}.\, N \bullet \vec{v} \Rrightarrow M}\ (\forall^{INF}_{\bullet\Rrightarrow})$$

$$\frac{\Theta; \Gamma \vdash v : P \quad \Theta \vdash Q \geqslant P \quad \Theta ; \Gamma \vdash N \bullet \vec{v} \Rrightarrow M}{\Theta ; \Gamma \vdash Q \to N \bullet v, \vec{v} \Rrightarrow M}\ (\to^{INF}_{\bullet\Rrightarrow})$$

Let us discuss the selected rules of the declarative system:

- $(\textsc{Var}^{\textsc{inf}})$ says that the type of a variable is inferred from the context.
- $(\{\}^{\textsc{inf}})$ says that the type of a thunk is inferred by shifting up the type of the contained computation. Symmetrically, $(\textsc{ret}^{\textsc{inf}})$ infers the type of a return by shifting down the type of the contained value.
- $(\textsc{ann}_{+}^{\textsc{inf}})$ and $(\textsc{ann}_{-}^{\textsc{inf}})$ are symmetric. They allow the inferred type to be refined by annotating it with a supertype.
- $(\simeq_{-}^{\textsc{inf}})$ and $(\simeq_{+}^{\textsc{inf}})$ mean that the declarative system allows us to infer any type from the equivalence class.
- $(\textsc{let}_{\exists}^{\textsc{inf}})$ is standard for existential types, and its first premise infers the existential type of the value being unpacked. It is important however that the inferred existential type is normalized. This is because there might be multiple equivalent existential types with a different order or even number of quantified variables, and to bind them, the algorithm needs to fix the canonical one.
- $(\textsc{let}_{:@}^{\textsc{inf}})$ allows us to type the *annotated* applicative let-binders. The first premise infers the type of the head of the application, which must be a thunked computation. Then if after applying it to the arguments, the resulting type can be instantiated to the annotated one, we infer the body of the let-binding in the context extended with the bound variable.
- $(\textsc{let}_{@}^{\textsc{inf}})$ is similar to $(\textsc{let}_{:@}^{\textsc{inf}})$, bus is used for unannotated let-bindings. In this case, we require the type application to infer the 'canonical' principal type. $\Theta\,;\Gamma \vdash M \bullet \vec{v} \Rightarrow > \uparrow Q$ principal means that any other type Q' inferrable for the application (i.e., $\Theta\,;\Gamma \vdash M \bullet \vec{v} \Rightarrow \uparrow Q'$) is greater than the principal type Q, i.e., $\Theta \vdash Q' \geqslant Q$.

Let us discuss the rules of the application inference:

- $(\emptyset_{\bullet\Rightarrow}^{\textsc{inf}})$ is the base case. If the list of arguments is empty, the inferred type is the type of the head. However, we relax this specification by allowing it to infer any other equivalent type. The relaxation of this rule is enough to guarantee this property for the whole judgement: if $\Theta\,;\Gamma \vdash N \bullet \vec{v} \Rightarrow M$ then $\Theta\,;\Gamma \vdash N \bullet \vec{v} \Rightarrow M'$ for any equivalent M'.
- $(\to_{\bullet\Rightarrow}^{\textsc{inf}})$ is where the application type is inferred: if the head has an arrow type $Q \to N$, we are allowed to apply it as soon as as soon as the first argument has a type, which is a subtype of Q.
- $(\forall_{\bullet\Rightarrow}^{\textsc{inf}})$ is the rule ensuring the implicit elimination of the universal quantifiers. If we are applying a polymorphic computation, we can instantiate its quantified variables with any types, which is expressed by the substitution $\Theta \vdash \sigma : \overrightarrow{\alpha^{\mp}}$.

A.4 Relation between $F_{\exists}^{\pm}$ and System F

Although based on System F, $F_{\exists}^{\pm}$ has an additional polarization structure. To demonstrate the relation between these systems we establish translations in both ways: the polarization

System F types			System F terms		
$T,\ A,\ B$	$::=$		$t,\ e$	$::=$	
	$\mid$	α		$\mid$	x
	$\mid$	$T_1 \to T_2$		$\mid$	$\lambda x.\, t$
	$\mid$	$\forall \vec{\alpha}.\, T$		$\mid$	$\Lambda \vec{\alpha}.\, t$
	$\mid$	$[T]$		$\mid$	$t_1\ t_2$
				$\mid$	$[\vec{t}\,]$
	$\mid$	$\exists \vec{\alpha}.\, T$		$\mid$	$t'\ \vec{t}$ $\quad \equiv ((t' t_1)\ \ldots)\ t_n$
		$\equiv \forall \beta.\, (\forall \vec{\alpha}.\, (T \to \beta)) \to \beta$		$\mid$	$\mathtt{let}\, x = t_1\, ;\, t_2$ $\quad \equiv (\lambda x.\, t_2)\ t_1$
				$\mid$	$\mathtt{pack}\, t\, \mathtt{as}\, T$ $\quad \equiv \Lambda \beta.\, \lambda f.\, f\ t$
				$\mid$	$\mathtt{unpack}\, (\vec{\alpha}, x) = t_1\, ;\, t_2$ $\quad \equiv t_1\ (\Lambda \vec{\alpha}.\, \lambda x.\, t_2)$

Fig. 25: Grammar of System F

$$\frac{x : T \in \Gamma}{\Theta; \Gamma \vdash x : T}\ (\textsc{Var}^{\mathsf{F}}) \qquad \frac{\Theta, \alpha; \Gamma \vdash t : T}{\Theta; \Gamma \vdash \Lambda\alpha.\, t : \forall \alpha.\, T}\ (\Lambda^{\mathsf{F}})$$

$$\frac{\Theta; \Gamma, x : A \vdash t : B}{\Theta; \Gamma \vdash \lambda x.\, t : A \to B}\ (\lambda^{\mathsf{F}}) \qquad \frac{\Theta; \Gamma \vdash t : \forall \alpha.\, T \quad \Theta \vdash A}{\Theta; \Gamma \vdash t : [A/\alpha]T}\ (\mathsf{TApp}^{\mathsf{F}})$$

$$\frac{\Theta; \Gamma \vdash t : A \to B \quad \Theta; \Gamma \vdash t' : A}{\Theta; \Gamma \vdash t\ t' : B}\ (\mathsf{App}^{\mathsf{F}})$$

Fig. 26: Typing rules of System F

from System F to $\mathrm{F}^{\pm}_{\exists}$ and the depolarization from $\mathrm{F}^{\pm}_{\exists}$ to System F. These translations are done at the level of types and terms, and the expected typing preservation properties are proved.

First, let us agree on the variant of System F that we use to establish the relation with $\mathrm{F}^{\pm}_{\exists}$. At the type level, we have variables, functional arrows, and universal quantifiers, and define existential quantifiers as a syntactic sugar using standard encodings. At the term level, we have variables, unannotated lambda abstractions, type abstractions, and term-level applications, but not type applications (they can be done implicitly); for convenience, we introduce multi-argument applications, let-blinders, and existential constructors and eliminators as syntactic sugar (see Fig. 25).

Observation 1. *The following rules are admissible in System F:*

$$\frac{\Theta; \Gamma \vdash t_1 : \exists \vec{\alpha}.\, T \quad \Theta, \vec{\alpha}; \Gamma, x : T \vdash t_2 : T' \quad \Theta \vdash T'}{\Theta; \Gamma \vdash \mathtt{unpack}\, (\vec{\alpha}, x) = t_1\, ;\, t_2 : T'}\ (\mathsf{Unpack}^{\mathsf{F}}) \qquad \frac{\Theta, \vec{\alpha} \vdash T \quad \Theta \vdash \vec{A} \quad \Theta; \Gamma \vdash t : [\vec{A}/\vec{\alpha}]T}{\Theta; \Gamma \vdash \mathtt{pack}\, t\, \mathtt{as}\, \exists \vec{\alpha}.\, T : \exists \vec{\alpha}.\, T}\ (\mathsf{Pack}^{\mathsf{F}})$$

$$\frac{\Theta; \Gamma \vdash t : A \quad \Theta; \Gamma, x : A \vdash t' : B}{\Theta; \Gamma \vdash \mathtt{let}\, x = t\, ;\, t' : B}\ (\mathsf{Let}^{\mathsf{F}})$$

$\boxed{|P|}$ $\boxed{|N|}$

$$|\alpha^+| \equiv \alpha \qquad |\alpha^-| \equiv \alpha$$
$$|{\downarrow}N| \equiv |N| \qquad |{\uparrow}P| \equiv |P|$$
$$|\exists \overrightarrow{\alpha^-}.\, P| \equiv \exists \overrightarrow{\alpha}.\, |P| \qquad |\forall \alpha^+.\, N| \equiv \forall \alpha.\, |N|$$
$$|P \to N| \equiv |P| \to |N|$$

Fig. 27: Type Depolarization

$\boxed{\lfloor T \rfloor}$

$$\lfloor \alpha \rfloor \equiv \alpha^+$$
$$\lfloor \forall \alpha.\, T \rfloor \equiv {\downarrow}(\forall \alpha^+.\, {\uparrow}\lfloor T \rfloor)$$
$$\lfloor A \to B \rfloor \equiv {\downarrow}(\lfloor A \rfloor \to {\uparrow}\lfloor B \rfloor)$$

Fig. 28: Type Polarization

A.4.1 Type-level Translation

The translation between the types is defined in Fig. 27 and Fig. 28.

The depolarization—translation from $\mathrm{F}^{\pm}_{\exists}$ to System F—is straightforward: it recursively (i) removes the shift operators $\uparrow$ and $\downarrow$, and (ii) removes the sign annotations from the type variables.

The polarization—translation from System F to $\mathrm{F}^{\pm}_{\exists}$—is more complex. There exist several ways to define it since any System F type can be polarized either positively or negatively. We chose the positive translation: every term of System F is translated into a value. Although this translation does not minimize the number of inserted shifts, it is more straightforward to consistently lift it to the level of terms.

A.4.2 Term-level Translation

The term-level translation is defined not solely for terms, but for typing derivations (we call this translation *elaboration*). The reason for that is that the terms of System F and $\mathrm{F}^{\pm}_{\exists}$ contain different typing information that cannot be reconstructed without the appropriate derivation tree. For instance, the lambda expressions in System F do not have the bound variable annotation, while the lambda expressions in $\mathrm{F}^{\pm}_{\exists}$ do; on the other hand, $\mathrm{F}^{\pm}_{\exists}$ has richer subtyping, which cannot be expressed in in terms of polymorphic type instantiation of System F.

The elaboration $\mathrm{F}^{\pm}_{\exists} \rightsquigarrow$ System F annotates each judgment of an $\mathrm{F}^{\pm}_{\exists}$ typing derivation tree with the corresponding System F term. This way, we define elaboration for each kind of judgment: subtyping, positive typing, negative typing, and application typing.

The subtyping elaboration is defined in Fig. 29. The soundness property that is preserved by the elaboration is stated in Lemma 52. Informally, $\Theta \vdash N \leqslant M \rightsquigarrow t$ guarantees that t is a System F term that represents a conversion (i.e., a function) from the depolarized $|N|$ to $|M|$. Symmetrically, $\Theta \vdash P \geqslant Q \rightsquigarrow t$ implies $|\Theta|; \cdot \vdash t : |Q| \to |P|$.

- Rules ($\textsc{Var}^{\rightsquigarrow}_{\leqslant}$) and ($\textsc{Var}^{\rightsquigarrow}_{\geqslant}$) are trivial: if the left-hand side and the right-hand side types are the same, the function converting one to another is the identity.
- Rules (${\uparrow}^{\rightsquigarrow}_{\leqslant}$) and (${\downarrow}^{\rightsquigarrow}_{\geqslant}$) are symmetric. Since the shifts are removed during the depolarization, the conversion function is obtained by the recursive call to the premise representing the subtyping of the required order (the mutual subtyping equivalence is defined as two separate judgments). Notice that the subtyping in the other order

$\boxed{\Theta \vdash N \leqslant M \rightsquigarrow t}$ $\boxed{\Theta \vdash P \geqslant Q \rightsquigarrow t}$

$$\frac{}{\Theta \vdash \alpha^- \leqslant \alpha^- \rightsquigarrow \lambda x.\, x}\ (\textsc{Var}^{\rightsquigarrow}_{\leqslant}) \qquad \frac{}{\Theta \vdash \alpha^+ \geqslant \alpha^+ \rightsquigarrow \lambda x.\, x}\ (\textsc{Var}^{\rightsquigarrow}_{\geqslant})$$

$$\frac{\Theta \vdash Q \geqslant P \rightsquigarrow t \quad \Theta \vdash P \geqslant Q}{\Theta \vdash \uparrow P \leqslant \uparrow Q \rightsquigarrow t}\ (\uparrow^{\rightsquigarrow}_{\leqslant}) \qquad \frac{\Theta \vdash N \leqslant M \quad \Theta \vdash M \leqslant N \rightsquigarrow t}{\Theta \vdash \downarrow N \geqslant \downarrow M \rightsquigarrow t}\ (\downarrow^{\rightsquigarrow}_{\geqslant})$$

$$\frac{\Theta \vdash P \geqslant Q \rightsquigarrow t \quad \Theta \vdash N \leqslant M \rightsquigarrow t'}{\Theta \vdash P \to N \leqslant Q \to M \rightsquigarrow \lambda x.\, \lambda y.\, t'\ (x\ (t\ y))}\ (\to^{\rightsquigarrow}_{\leqslant})$$

$$\frac{\Theta, \overrightarrow{\beta^-} \vdash \sigma : \overrightarrow{\alpha^-} \quad \Theta, \overrightarrow{\beta^-} \vdash [\sigma]P \geqslant Q \rightsquigarrow t}{\Theta \vdash \exists \overrightarrow{\alpha^-}.\, P \geqslant \exists \overrightarrow{\beta^-}.\, Q \rightsquigarrow \lambda x.\, \mathsf{unpack}\ (\overrightarrow{\beta}, y) = x\,;\, \mathsf{pack}\ (t\ y)\ \mathsf{as}\ \exists \overrightarrow{\alpha}.\, |P|}\ (\exists^{\rightsquigarrow}_{\geqslant})$$

$$\frac{\Theta, \overrightarrow{\beta^+} \vdash \sigma : \overrightarrow{\alpha^+} \quad \Theta, \overrightarrow{\beta^+} \vdash [\sigma]N \leqslant M \rightsquigarrow t}{\Theta \vdash \forall \overrightarrow{\alpha^+}.\, N \leqslant \forall \overrightarrow{\beta^+}.\, M \rightsquigarrow \lambda x.\, \Lambda \overrightarrow{\beta}.\, t\ x}\ (\forall^{\rightsquigarrow}_{\leqslant})$$

Fig. 29: Subtyping elaboration from $\mathrm{F}^{\pm}_{\exists}$ to System F

is still required so that the removal of the elaboration produces a correct subtyping inference tree in the original $\mathrm{F}^{\pm}_{\exists}$.

- Rule $(\to^{\rightsquigarrow}_{\leqslant})$ allows us to define elaboration between function. Since the subtyping is contravariant on the argument type and covariant on the result type, we can use the elaboration functions acquired in the premises (their types are $|Q| \to |P|$ and $|N| \to |M|$) to construct the required elaboration term of type $(|P| \to |N|) \to |Q| \to |M|$.
- Rules $(\forall^{\rightsquigarrow}_{\leqslant})$, and $(\exists^{\rightsquigarrow}_{\geqslant})$ are more involved. However, the soundness property is preserved by Observation 4 allowing one to distribute depolarization over the substitution whereby translating the instantiation substitution σ from $\mathrm{F}^{\pm}_{\exists}$ to System F.

Using the subtyping elaboration, we define the elaboration for the positive/negative typing and application typing in Fig. 30. The soundness property (Lemma 53) guarantees that if the initial term v has type P in $\mathrm{F}^{\pm}_{\exists}$, then the result of the elaboration of this judgment t has type $|P|$ in System F. The negative typing has a symmetric property. As with the subtyping elaboration, the rules in Fig. 30 are obtained from the $\mathrm{F}^{\pm}_{\exists}$ declarative typing rules (Definition 28) by annotating each judgment with the elaboration term so that the typing is preserved.

$\boxed{\Theta; \Gamma \vdash c : N \rightsquigarrow t}$ Negative typing elaboration

$$\frac{\Theta \vdash P \quad \Theta; \Gamma, x : P \vdash c : N \rightsquigarrow t}{\Theta; \Gamma \vdash \lambda x : P.\, c : P \to N \rightsquigarrow \lambda x.\, t}\ (\lambda^{\rightsquigarrow}) \qquad \frac{\Theta, \alpha^{+}; \Gamma \vdash c : N \rightsquigarrow t}{\Theta; \Gamma \vdash \Lambda \alpha^{+}.\, c : \forall \alpha^{+}.\, N \rightsquigarrow \Lambda \alpha.\, t}\ (\Lambda^{\rightsquigarrow})$$

$$\frac{\Theta; \Gamma \vdash v : P \rightsquigarrow t}{\Theta; \Gamma \vdash \mathtt{return}\, v : \uparrow P \rightsquigarrow t}\ (\mathrm{RET}^{\rightsquigarrow}) \qquad \frac{\Theta; \Gamma \vdash v : P \rightsquigarrow t \quad \Theta; \Gamma, x : P \vdash c : N \rightsquigarrow t'}{\Theta; \Gamma \vdash \mathtt{let}\, x = v;\ c : N \rightsquigarrow \mathtt{let}\, x = t\, ;\, t'}\ (\mathrm{LET}^{\rightsquigarrow})$$

$$\frac{\Theta; \Gamma \vdash c : N \rightsquigarrow t \quad \Theta \vdash N' \leqslant N \quad \Theta \vdash N \leqslant N' \rightsquigarrow e}{\Theta; \Gamma \vdash c : N' \rightsquigarrow e\ t}\ (\simeq^{\rightsquigarrow}_{-}) \qquad \frac{\Theta \vdash M \quad \Theta; \Gamma \vdash c : N \rightsquigarrow t \quad \Theta \vdash N \leqslant M \rightsquigarrow e}{\Theta; \Gamma \vdash (c : M) : M \rightsquigarrow e\ t}\ (\mathrm{ANN}^{\rightsquigarrow}_{-})$$

$$\frac{\Theta; \Gamma \vdash v : \downarrow M \rightsquigarrow t' \quad \Theta\, ; \Gamma \vdash M \bullet \vec{v} \Rrightarrow \uparrow Q \text{ principal} \quad \Theta; \Gamma \vdash M \bullet \vec{v} \Rrightarrow \uparrow Q \rightsquigarrow e;\, \vec{t} \quad \Theta; \Gamma, x : Q \vdash c : N \rightsquigarrow t}{\Theta; \Gamma \vdash \mathtt{let}\, x = v(\vec{v});\ c : N \rightsquigarrow \mathtt{let}\, x = (e\ (t'\ \vec{t}))\, ;\, t}\ (\mathrm{LET}^{\rightsquigarrow}_{@})$$

$$\frac{\Theta \vdash P \quad \Theta; \Gamma \vdash v : \downarrow M \rightsquigarrow t' \quad \Theta; \Gamma \vdash M \bullet \vec{v} \Rrightarrow M' \rightsquigarrow e;\, \vec{t} \quad \Theta \vdash M' \leqslant \uparrow P \rightsquigarrow e' \quad \Theta; \Gamma, x : P \vdash c : N \rightsquigarrow t}{\Theta; \Gamma \vdash \mathtt{let}\, x : P = v(\vec{v});\ c : N \rightsquigarrow \mathtt{let}\, x = e'\ (e\ (t'\ \vec{t}))\, ;\, t}\ (\mathrm{LET}^{\rightsquigarrow}_{:@})$$

$$\frac{\Theta; \Gamma \vdash v : \exists \overrightarrow{\alpha}.\, P \rightsquigarrow t \quad \mathtt{nf}\,(\exists \overrightarrow{\alpha}.\, P) = \exists \overrightarrow{\alpha}.\, P \quad \Theta, \overrightarrow{\alpha}; \Gamma, x : P \vdash c : N \rightsquigarrow t' \quad \Theta \vdash N}{\Theta; \Gamma \vdash \mathtt{let}^{\exists}(\overrightarrow{\alpha}, x) = v;\ c : N \rightsquigarrow \mathtt{unpack}\,(\overrightarrow{\alpha}, x) = t\, ;\, t'}\ (\mathrm{LET}^{\rightsquigarrow}_{\exists})$$

$$\frac{\Theta \vdash P \quad \Theta; \Gamma \vdash c : M \rightsquigarrow t \quad \Theta \vdash M \leqslant \uparrow P \rightsquigarrow e \quad \Theta; \Gamma, x : P \vdash c' : N \rightsquigarrow t'}{\Theta; \Gamma \vdash \mathtt{let}\, x : P = c;\ c' : N \rightsquigarrow \mathtt{let}\, x = (e\ t)\, ;\, t'}\ (\mathrm{LET}^{\rightsquigarrow}_{\mathrm{C}})$$

$\boxed{\Theta; \Gamma \vdash v : P \rightsquigarrow t}$
Positive typing elaboration

$$\frac{x : P \in \Gamma}{\Theta; \Gamma \vdash x : P \rightsquigarrow x}\ (\mathrm{VAR}^{\rightsquigarrow}) \qquad \frac{\Theta; \Gamma \vdash c : N \rightsquigarrow t}{\Theta; \Gamma \vdash \{c\} : \downarrow N \rightsquigarrow t}\ (\{\}^{\rightsquigarrow})$$

$$\frac{\Theta \vdash Q \quad \Theta; \Gamma \vdash v : P \rightsquigarrow t \quad \Theta \vdash Q \geqslant P \rightsquigarrow e}{\Theta; \Gamma \vdash (v : Q) : Q \rightsquigarrow e\ t}\ (\mathrm{ANN}^{\rightsquigarrow}_{+}) \qquad \frac{\Theta; \Gamma \vdash v : P \rightsquigarrow t \quad \Theta \vdash P \geqslant P' \quad \Theta \vdash P' \geqslant P \rightsquigarrow e}{\Theta; \Gamma \vdash v : P' \rightsquigarrow e\ t}\ (\simeq^{\rightsquigarrow}_{+})$$

$\boxed{\Theta; \Gamma \vdash N \bullet \vec{v} \Rrightarrow M \rightsquigarrow e;\, \vec{t}}$
Application typing

$$\frac{\Theta \vdash N \leqslant N' \quad \Theta \vdash N' \leqslant N \rightsquigarrow e}{\Theta; \Gamma \vdash N \bullet \Rrightarrow N' \rightsquigarrow e; \cdot}\ (\emptyset^{\rightsquigarrow}_{\bullet\Rrightarrow})$$

$$\frac{\Theta; \Gamma \vdash v : P \rightsquigarrow t \quad \Theta \vdash Q \geqslant P \rightsquigarrow e \quad \Theta; \Gamma \vdash N \bullet \vec{v} \Rrightarrow M \rightsquigarrow e';\, \vec{t}}{\Theta; \Gamma \vdash Q \to N \bullet v,\, \vec{v} \Rrightarrow M \rightsquigarrow e'; e\ t,\, \vec{t}}\ (\to^{\rightsquigarrow}_{\bullet\Rrightarrow})$$

$$\frac{\vec{v} \neq \cdot \quad \overrightarrow{\alpha^{+}} \neq \cdot \quad \Theta \vdash \sigma : \overrightarrow{\alpha^{+}} \quad \Theta; \Gamma \vdash [\sigma]N \bullet \vec{v} \Rrightarrow M \rightsquigarrow e;\, \vec{t}}{\Theta; \Gamma \vdash \forall \overrightarrow{\alpha^{+}}.\, N \bullet \vec{v} \Rrightarrow M \rightsquigarrow e;\, \vec{t}}\ (\forall^{\rightsquigarrow}_{\bullet\Rrightarrow})$$

Fig. 30: Typing elaboration from $\mathrm{F}^{\pm}_{\exists}$ to System F

$\boxed{\Theta; \Gamma \vdash t : T \rightsquigarrow^{\pm} c}$

$$\frac{x : T \in \Gamma}{\Theta; \Gamma \vdash x : T \rightsquigarrow^{\pm} \mathtt{return}\, x}\ (\textsc{Var}^{\rightsquigarrow^{\pm}})$$

$$\frac{\Theta; \Gamma, x : A \vdash t : B \rightsquigarrow^{\pm} c}{\Theta; \Gamma \vdash \lambda x.\, t : A \to B \rightsquigarrow^{\pm} \mathtt{return}\, \{\lambda x : \lfloor A \rfloor .\, c\}}\ (\lambda^{\rightsquigarrow^{\pm}})$$

$$\frac{\Theta; \Gamma \vdash t : A \to B \rightsquigarrow^{\pm} c \quad \Theta; \Gamma \vdash t' : A \rightsquigarrow^{\pm} c'}{\Theta; \Gamma \vdash t\, t' : B \rightsquigarrow^{\pm} \mathtt{let}\, f : \lfloor A \to B \rfloor = c;\ \mathtt{let}\, x : \lfloor A \rfloor = c';\ \mathtt{let}\, y : \lfloor B \rfloor = f(x);\ \mathtt{return}\, y}\ (\mathsf{App}^{\rightsquigarrow^{\pm}})$$

$$\frac{\Theta, \alpha; \Gamma \vdash t : T \rightsquigarrow^{\pm} c}{\Theta; \Gamma \vdash \Lambda\alpha.\, t : \forall \alpha.\, T \rightsquigarrow^{\pm} \mathtt{return}\, \{\Lambda \alpha^{+} .\, c\}}\ (\Lambda^{\rightsquigarrow^{\pm}})$$

$$\frac{}{\Theta; \Gamma \vdash t : [A/\alpha]T \rightsquigarrow^{\pm} \mathtt{let}\, f : \lfloor \forall \alpha.\, T \rfloor = c;\ \mathtt{let}\, y : \lfloor [A/\alpha]T \rfloor = f();\ \mathtt{return}\, y}\ (\mathsf{TApp}^{\rightsquigarrow^{\pm}})$$

Fig. 31: Typing elaboration form from System F to $\mathrm{F}^{\pm}_{\exists}$

The other direction of translation is also represented as an elaboration. The inference rules are defined in Fig. 31, and constitute the rules of System F (Fig. 26). annotated with the elaboration terms. The elaborated term is chosen in such a way that the soundness property (Lemma 56) holds: $\Theta; \Gamma \vdash t : T \rightsquigarrow^{\pm} c$ implies $\lfloor \Theta \rfloor; \lfloor \Gamma \rfloor \vdash c : \uparrow \lfloor T \rfloor$.

A.5 Algorithmic Typing

Next, we present the type inference algorithm, which is sound and complete with respect to the declarative specification (Definition 28).

A.5.1 Algorithmic Type Inference

Mirroring the declarative typing, the algorithm is represented by an inference system of three mutually recursive judgments:

- $\Theta; \Gamma \vDash v : P$ and $\Theta; \Gamma \vDash c : N$ are the algorithmic versions of $\Theta; \Gamma \vdash v : P$ and $\Theta; \Gamma \vdash c : N$. In contrast with the declarative counterparts, they are deterministic, and guarantee that the inferred type is normalized.
- $\Theta\,; \Gamma\,; \Xi_1 \vDash N \bullet \vec{v} \Longrightarrow M \dashv \Xi_2\,; C$ is the algorithmization of $\Theta\,; \Gamma \vdash N \bullet \vec{v} \Longrightarrow M$. Notice that N contains algorithmic variables, which are specified by the context Ξ_1. Moreover, the inferred type M is also algorithmic, and can have several non-equivalent instantiations. To accommodate that, the algorithm also returns Ξ_2 and C specifying the variables used in M: Ξ_2 defines the contexts in which the variables must be instantiated, and C imposes restrictions on the variables.

As subroutines, the algorithm calls subtyping (Algorithm 7), type well-formedness (Algorithm 1), constraint merge (Section A.2.8), normalization (Algorithm 5), and constraint singularity which will be defined later in Section A.5.3. It also relies on basic set operations and the ability to deterministically choose fresh variables.

Algorithm 14.

$\boxed{\Theta; \Gamma \vDash v \colon P}$ *Positive typing*

$$\frac{x : P \in \Gamma}{\Theta; \Gamma \vDash x \colon \mathtt{nf}\,(P)}\ (\textsc{Var}^{\textsc{inf}}) \qquad \frac{\Theta \vdash Q \quad \Theta; \Gamma \vDash v \colon P \quad \Theta; \cdot \vDash Q \geqslant P \dashv \cdot}{\Theta; \Gamma \vDash (v : Q) \colon \mathtt{nf}\,(Q)}\ (\textsc{ann}_{+}^{\textsc{inf}}) \qquad \frac{\Theta; \Gamma \vDash c \colon N}{\Theta; \Gamma \vDash \{c\} \colon \downarrow N}\ (\{\}^{\textsc{inf}})$$

$\boxed{\Theta; \Gamma \vDash c \colon N}$ *Negative typing*

$$\frac{\Theta \vdash M \quad \Theta; \Gamma \vDash c \colon N \quad \Theta; \cdot \vDash N \leqslant M \dashv \cdot}{\Theta; \Gamma \vDash (c : M) \colon \mathtt{nf}\,(M)}\ (\textsc{ann}_{-}^{\textsc{inf}}) \qquad \frac{\Theta; \Gamma \vDash v \colon P \quad \Theta; \Gamma, x : P \vDash c \colon N}{\Theta; \Gamma \vDash \mathtt{let}\, x = v;\ c \colon N}\ (\textsc{let}^{\textsc{inf}})$$

$$\frac{\Theta \vdash P \quad \Theta; \Gamma, x : P \vDash c \colon N}{\Theta; \Gamma \vDash \lambda x : P.\, c \colon \mathtt{nf}\,(P \to N)}\ (\lambda^{\textsc{inf}}) \qquad \frac{\Theta \vdash P \quad \Theta; \Gamma \vDash c \colon M \quad \Theta; \cdot \vDash M \leqslant \uparrow P \dashv \cdot \quad \Theta; \Gamma, x : P \vDash c' \colon N}{\Theta; \Gamma \vDash \mathtt{let}\, x : P = c;\ c' \colon N}\ (\textsc{let}_{C}^{\textsc{inf}})$$

$$\frac{\Theta, \alpha^{+}; \Gamma \vDash c \colon N}{\Theta; \Gamma \vDash \Lambda \alpha^{+}.\, c \colon \mathtt{nf}\,(\forall \alpha^{+}.\, N)}\ (\Lambda^{\textsc{inf}}) \qquad \frac{\Theta; \Gamma \vDash v \colon \exists \overrightarrow{\alpha^{+}}.\, P \quad \Theta, \overrightarrow{\alpha^{+}}; \Gamma, x : P \vDash c \colon N \quad \Theta \vdash N}{\Theta; \Gamma \vDash \mathtt{let}^{\exists}(\overrightarrow{\alpha^{+}}, x) = v;\ c \colon N}\ (\textsc{let}_{\exists}^{\textsc{inf}})$$

$$\frac{\Theta; \Gamma \vDash v \colon P}{\Theta; \Gamma \vDash \mathtt{return}\, v \colon \uparrow P}\ (\textsc{ret}^{\textsc{inf}})$$

$$\frac{\Theta \vdash P \quad \Theta; \Gamma \vDash v \colon \downarrow M \quad \Theta\,; \Gamma\,; \cdot \vDash M \bullet \vec{v} \Rrightarrow M' \dashv \Xi\,; C_1 \quad \Theta; \Xi \vDash M' \leqslant \uparrow P \dashv C_2 \quad \Xi \vdash C_1 \,\&\, C_2 = C \quad \Theta; \Gamma, x : P \vDash c \colon N}{\Theta; \Gamma \vDash \mathtt{let}\, x : P = v(\vec{v});\ c \colon N}\ (\textsc{let}_{:@}^{\textsc{inf}})$$

$$\frac{\Theta; \Gamma \vDash v \colon \downarrow M \quad \Theta\,; \Gamma\,; \cdot \vDash M \bullet \vec{v} \Rrightarrow \uparrow Q \dashv \Xi\,; C \quad Q \text{ is } C\text{-minimized by } \widehat{\sigma} \quad \Theta; \Gamma, x : [\widehat{\sigma}]Q \vDash c \colon N}{\Theta; \Gamma \vDash \mathtt{let}\, x = v(\vec{v});\ c \colon N}\ (\textsc{let}_{@}^{\textsc{inf}})$$

$\boxed{\Theta\,; \Gamma\,; \Xi_1 \vDash N \bullet \vec{v} \Rrightarrow M \dashv \Xi_2\,; C}$ *Application typing*

$$\frac{}{\Theta\,; \Gamma\,; \Xi \vDash N \bullet \Rrightarrow \mathtt{nf}\,(N) \dashv \Xi\,; \cdot}\ (\emptyset_{\bullet\Rrightarrow}^{\textsc{inf}}) \qquad \frac{\Theta; \Gamma \vDash v \colon P \quad \Theta; \Xi \vDash Q \geqslant P \dashv C_1 \quad \Theta\,; \Gamma\,; \Xi \vDash N \bullet \vec{v} \Rrightarrow M \dashv \Xi'\,; C_2 \quad \Xi \vdash C_1 \,\&\, C_2 = C}{\Theta\,; \Gamma\,; \Xi \vDash Q \to N \bullet v, \vec{v} \Rrightarrow M \dashv \Xi'\,; C}\ (\to_{\bullet\Rrightarrow}^{\textsc{inf}})$$

$$\frac{\Theta\,; \Gamma\,; \Xi, \overrightarrow{\widehat{\alpha^{+}}}\{\Theta\} \vDash [\overrightarrow{\widehat{\alpha^{+}}}/\overrightarrow{\alpha^{+}}]N \bullet \vec{v} \Rrightarrow M \dashv \Xi'\,; C \quad \overrightarrow{\widehat{\alpha^{+}}} \text{ are fresh} \quad \vec{v} \neq \quad \overrightarrow{\alpha^{+}} \neq \cdot}{\Theta\,; \Gamma\,; \Xi \vDash \forall \overrightarrow{\alpha^{+}}.\, N \bullet \vec{v} \Rrightarrow M \dashv \Xi'\,; C|_{\mathtt{fav}(N) \cup \mathtt{fav}(M)}}\ (\forall_{\bullet\Rrightarrow}^{\textsc{inf}})$$

Let us discuss the inference rules of the algorithm:

- ($\textsc{Var}^{\textsc{inf}}$) infers the type of a variable by looking it up in the context and normalizing the result.

- $(\{\}^{\text{INF}})$ and $(\text{RET}^{\text{INF}})$ are similar to the declarative rules: they make a recursive call to type the body of the thunk or the return expression and put the shift on top of the result.
- $(\text{ANN}_{+}^{\text{INF}})$ and $(\text{ANN}_{-}^{\text{INF}})$ are symmetric. They make a recursive call to infer the type of the annotated expression, check that the inferred type is a subtype of the annotation, and return the normalized annotation.
- (λ^{INF}) infers the type of a lambda-abstraction. It makes a recursive call to infer the type of the body in the extended context, and returns the corresponding arrow type. Notice that the algorithm also normalizes the result, which is because the annotation type P is allowed to be non-normalized.
- (Λ^{INF}) infers the type of a big lambda. Similarly to the previous case, it makes a recursive call to infer the type of the body in the extended *type* context. After that, it returns the corresponding universal type. It is also required to normalize the result, because, for instance, α^{+} might not occur in the body of the lambda, in which case the $\forall$ must be removed.
- $(\text{LET}^{\text{INF}})$ is defined in a standard way: it makes a recursive call to infer the type of the bound value, and then returns the type of the body in the extended context.
- $(\text{LET}_{:@}^{\text{INF}})$ is interpreted as follows. First, it infers the type of the head of the application, ensuring that it is a thunked computation $\downarrow M$; after that, it makes a recursive call to the application inference procedure, which returns the algorithmic type, whose instantiation to a declarative type must be associated with the bound variable x; then premise $\Theta;\ \Xi \vDash M' \leqslant \uparrow P \dashv C_2$ together with $\Xi \vdash C_1 \,\&\, C_2 = C$ check whether the instantiation to the annotated type P is possible, and if it is, the algorithm infers the type of the body in the extended context, and returns it as the result.
- $(\text{LET}_{@}^{\text{INF}})$ works similarly to $(\text{LET}_{:@}^{\text{INF}})$, However, since there is no annotation to assign the result to, the algorithm must infer the 'canonical' principal type. To do that algorithmically, we ensure that the inferred algorithmic type Q is instantiated to the minimal possible type $[\widehat{\sigma}]\,Q$. The premise Q is C-minimized by $\widehat{\sigma}$ provides the minimal instantiation of Q w.r.t. C. It guarantees that if we consider all possible substitutions satisfying the inferred constraints C, then substitution $\widehat{\sigma}$ will instantiate Q to the minimal possible type $[\widehat{\sigma}]\,Q$. This will be the principal type that we assign to the result of the application (bound to the variable x) and then we infer the type of the body in the context extended with the bound variable $x : [\widehat{\sigma}]\,Q$.
- $(\text{LET}_{\exists}^{\text{INF}})$ first, infers the existential type $\exists\overrightarrow{\alpha^{-}}.\ P$ of the value being unpacked, and since the type is guaranteed to be normalized, binds the quantified variables with $\overrightarrow{\alpha^{-}}$. Then it infers the type of the body in the appropriately extended context and checks that the inferred type does not depend on $\overrightarrow{\alpha^{-}}$ by checking well-formedness $\Theta \vdash N$.

Finally, let us discuss the algorithmic rules of the application inference:

- $(\emptyset_{\bullet\Rightarrow}^{\text{INF}})$ is the base case. If the list of arguments is empty, the inferred type is the type of the head, and the algorithm returns it after normalizing.

- $(\to^{\mathrm{INF}}_{\bullet\Rightarrow})$ is the main rule of algorithmic application inference. If the head has an arrow type $Q \to N$, we find C_1—the minimal constraint ensuring that Q is a supertype of the first argument's type. Then we make a recursive call applying N to the rest of the arguments, and merge the resulting constraint with C_1
- $(\forall^{\mathrm{INF}}_{\bullet\Rightarrow})$, analogously to the declarative case, is the rule ensuring the implicit elimination of the universal quantifiers. This is the place where the algorithmic variables are generated. The algorithm simply replaces the quantified variables $\overrightarrow{\alpha^+}$ with fresh algorithmic variables $\overrightarrow{\widehat{\alpha}^+}$, and makes a recursive call in the extended context.

The correctness of the algorithm consists of its soundness and completeness, which is proved by mutual induction in Lemmas 101 and 102. The simplified result is the following.

Theorem (Correctness of Algorithmic Typing (simplified))**.**

- $\Theta; \Gamma \vDash c : N$ *implies* $\Theta; \Gamma \vdash c : N$, *and* $\Theta; \Gamma \vdash c : N$ *implies* $\Theta; \Gamma \vDash c : \mathtt{nf}\,(N)$;
- $\Theta; \Gamma \vDash v : P$ *implies* $\Theta; \Gamma \vdash v : P$, *and* $\Theta; \Gamma \vdash v : P$ *implies* $\Theta; \Gamma \vDash v : \mathtt{nf}\,(P)$.

A.5.2 Minimal Instantiation

The minimal instantiation algorithm is used to infer the type of the bound variable in the un-annotated applicative let-binders, as long as there exists a principal (minimal) type. Given a positive algorithmic type P and a set of constraints C, it finds the substitution $\widehat{\sigma}$ respecting C (i.e., $\Xi \vdash \widehat{\sigma} : C$) such that it instantiates P to the minimal type, in other words for any other substitution $\widehat{\sigma}'$ respecting C, we have $\Theta \vdash [\widehat{\sigma}']P \geqslant [\widehat{\sigma}]P$.

The minimal instantiation algorithm is defined as follows:
Algorithm 15 (Minimal Instantiation)**.**
P *is* C*-minimized by* $\widehat{\sigma}$

$$\frac{(\widehat{\alpha}^+ :\geqslant P) \in C}{\widehat{\alpha}^+ \text{ is } C\text{-minimized by } (\mathtt{nf}\,(P)/\widehat{\alpha}^+)}\ (UVAR^{MIN})$$

$$\frac{P \text{ is } C\text{-minimized by } \widehat{\sigma}}{\exists\overrightarrow{\widehat{\alpha}}.\ P \text{ is } C\text{-minimized by } \widehat{\sigma}}\ (\exists^{MIN})$$

$$\frac{\mathtt{fav}(P) \subseteq \mathtt{dom}\,(C) \qquad C|_{\mathtt{fav}(P)}\ \mathtt{singular\ with}\ \widehat{\sigma}}{P \text{ is } C\text{-minimized by } \widehat{\sigma}}\ (SING^{MIN})$$

A.5.3 Constraint Singularity

The singularity algorithm checks whether the constraint C uniquely defines the substitution satisfying it, and if it does, the algorithm returns this substitution as the result. To implement it, we define a partial function $C\ \mathtt{singular\ with}\ \widehat{\sigma}$, taking a subtyping constraint C as an argument and returning a substitution $\widehat{\sigma}$—the only possible solution of C.

First, we define the notion of singularity on constraint entries. e `singular with` P and its negative counterpart are considered partial functions taking a constraint entry e and returning the type satisfying e if such a type is unique.

Algorithm 16 (Singular Constraint Entry)**.**

e `singular with` P

$$\frac{}{\widehat{\alpha}^+ :\simeq P \texttt{ singular with nf } (P)}\ (\simeq_+^{SING})$$

$$\frac{}{\widehat{\alpha}^+ :\geqslant \exists\overrightarrow{\alpha^-}.\ \alpha^+ \texttt{ singular with } \alpha^+}\ (:\geqslant\alpha^{SING})$$

$$\frac{\texttt{nf } (N) = \alpha_i^- \in \overrightarrow{\alpha^-}}{\widehat{\alpha}^+ :\geqslant \exists\overrightarrow{\alpha^-}.\downarrow N \texttt{ singular with } \exists\beta^-.\downarrow\beta^-}\ (:\geqslant\downarrow^{SING})$$

e `singular with` N

$$\frac{}{\widehat{\alpha}^- :\simeq N \texttt{ singular with nf } (N)}\ (\simeq_-^{SING})$$

- $(\simeq_-^{SING})$ and $(\simeq_+^{SING})$ are symmetric. If the constraint entry says that a variable must be equivalent to a type T, then it is evidently singular, and the only (up-to-equivalence) type instantiating this variable could be T. This way, we return its normal form.
- $(:\geqslant\alpha^{SING})$ implies that the only (normalized) solution of $\widehat{\alpha}^+ :\geqslant \exists\overrightarrow{\alpha^-}.\ \alpha^+$ is α^+ (it will be shown in Lemma 19).
- $(:\geqslant\downarrow^{SING})$ is perhaps the least obvious rule. Having a type $\exists\overrightarrow{\alpha^-}.\downarrow N$, suppose that N is not equivalent to any just bound variable ${\alpha^-}_i \in \overrightarrow{\alpha^-}$. Then the type $\exists\overrightarrow{\alpha^-}.\downarrow N$ has a proper supertype: $\exists\overrightarrow{\alpha^-}.\downarrow{\alpha^-}_1$, and thus the constraint is not singular. Otherwise, if N is equivalent to some ${\alpha^-}_i$, any supertype of $\exists\overrightarrow{\alpha^-}.\downarrow{\alpha^-}_i$ is equivalent to it, and thus, the constraint has a unique solution.

Next, we extrapolate the singularity function on constraints—sets of constraint entries. We require C to be a set of singular constraints, and the resulting substitution sends each variable from $\texttt{dom}\,(C)$ to the unique type satisfying the corresponding constraint.

Algorithm 17. *C* `singular with` *$\widehat{\sigma}$ means that*

1. *for any positive $e \in C$, there exists P such that e* `singular with` *P, and for any negative $e \in C$, there exists N such that e* `singular with` *N;*
2. *$\widehat{\sigma}$ is defined as follows:*

$$[\widehat{\sigma}]\widehat{\beta}^+ = \begin{cases} P & \text{if there is } e \in \texttt{dom}\,(C) \text{ restricting } \widehat{\beta}^+ \text{ and } e \texttt{ singular with } P \\ \widehat{\beta}^+ & \text{otherwise} \end{cases}$$

$$[\widehat{\sigma}]\widehat{\beta}^- = \begin{cases} N & \text{if there is } e \in \texttt{dom}\,(C) \text{ restricting } \widehat{\beta}^- \text{ and } e \texttt{ singular with } N \\ \widehat{\beta}^- & \text{otherwise} \end{cases}$$

The correctness of the singularity algorithm is formulated as follows:

Theorem. *Suppose that C is a subtyping constraint. Then C* $\texttt{singular with}\ \widehat{\sigma}$ *holds if and only if $\widehat{\sigma}$ is the only (up-to-equivalence on* $\texttt{dom}\,(C)$*) normalized substitution satisfying C.*

Appendix B Theorem Statements

B.1 Theorem Statements: Declarative

B.1.1 Type Well-Formedness

Lemma 3 (Soundness of type well-formedness)**.**

+ *If* $\Theta \vdash P$ *then* $\mathtt{fv}\ (P) \subseteq \Theta$,
− *if* $\Theta \vdash N$ *then* $\mathtt{fv}\ (N) \subseteq \Theta$.

Lemma 4 (Completeness of type well-formedness)**.** *In the well-formedness judgment, only used variables matter:*

+ *if* $\Theta_1 \cap \mathtt{fv}\ P = \Theta_2 \cap \mathtt{fv}\ P$ *then* $\Theta_1 \vdash P \iff \Theta_2 \vdash P$,
− *if* $\Theta_1 \cap \mathtt{fv}\ N = \Theta_2 \cap \mathtt{fv}\ N$ *then* $\Theta_1 \vdash N \iff \Theta_2 \vdash N$.

Corollary 1 (Context Strengthening)**.**

+ *If* $\Theta \vdash P$ *then* $\mathtt{fv}\ (P) \vdash P$;
− *If* $\Theta \vdash N$ *then* $\mathtt{fv}\ (N) \vdash N$.

Corollary 2 (Well-formedness Context Weakening)**.** *Suppose that* $\Theta_1 \subseteq \Theta_2$, *then*

+ *if* $\Theta_1 \vdash P$ *then* $\Theta_2 \vdash P$,
− *if* $\Theta_1 \vdash N$ *then* $\Theta_2 \vdash N$.

Lemma 5 (Well-formedness agrees with substitution)**.** *Suppose that* $\Theta_2 \vdash \sigma : \Theta_1$. *Then*

+ $\Theta, \Theta_1 \vdash P$ *implies* $\Theta, \Theta_2 \vdash [\sigma]P$, *and*
− $\Theta, \Theta_1 \vdash N$ *implies* $\Theta, \Theta_2 \vdash [\sigma]N$.

B.1.2 Substitution

Lemma 6 (Substitution strengthening)**.** *Restricting the substitution to the free variables of the substitution subject does not affect the result. Suppose that* σ *is a substitution,* P *and* N *are types. Then*

+ $[\sigma]P = [\sigma|_{\mathtt{fv}\ P}]P$,
− $[\sigma]N = [\sigma|_{\mathtt{fv}\ N}]N$

Lemma 7 (Signature of a restricted substitution)**.** *If* $\Theta_2 \vdash \sigma : \Theta_1$ *then* $\Theta_2 \vdash \sigma|_{vars} : \Theta_1 \cap vars$.

Lemma 8. *Suppose that* σ *is a substitution with signature* $\Theta_2 \vdash \sigma : \Theta_1$. *Then if vars is disjoint from* Θ_1, *then* $\sigma|_{vars} = \mathtt{id}$.

Corollary 3 (Application of a disjoint substitution)**.** *Suppose that* σ *is a substitution with signature* $\Theta_2 \vdash \sigma : \Theta_1$. *Then*

+ *if* $\Theta_1 \cap \mathtt{fv}\ (Q) = \emptyset$ *then* $[\sigma]Q = Q$;
− *if* $\Theta_1 \cap \mathtt{fv}\ (N) = \emptyset$ *then* $[\sigma]N = N$.

Lemma 9 (Substitution range weakening)**.** *Suppose that* $\Theta_2 \subseteq \Theta_2'$ *are contexts and* σ *is a substitution. Then* $\Theta_2 \vdash \sigma : \Theta_1$ *implies* $\Theta_2' \vdash \sigma : \Theta_1$.

Lemma 10 (Subsitutions Equivalent on Free Variables)**.** *Suppose that* $\Theta' \subseteq \Theta$*,* σ_1 *and* σ_2 *are substitutions of signature* $\Theta \vdash \sigma_i : \Theta'$*. Then*

+ *for a type* $\Theta \vdash P$*, if* $\Theta \vdash [\sigma_1]P \simeq^{\leqslant} [\sigma_2]P$ *then* $\Theta \vdash \sigma_1 \simeq^{\leqslant} \sigma_2 : \mathtt{fv}\ P \cap \Theta'$*;*

− *for a type* $\Theta \vdash N$*, if* $\Theta \vdash [\sigma_1]N \simeq^{\leqslant} [\sigma_2]N$ *then* $\Theta \vdash \sigma_1 \simeq^{\leqslant} \sigma_2 : \mathtt{fv}\ N \cap \Theta'$*.*

Lemma 11 (Substitution composition well-formedness)**.** *If* $\Theta'_1 \vdash \sigma_1 : \Theta_1$ *and* $\Theta'_2 \vdash \sigma_2 : \Theta_2$*, then* $\Theta'_1, \Theta'_2 \vdash \sigma_2 \circ \sigma_1 : \Theta_1, \Theta_2$*.*

Lemma 12 (Substitution monadic composition well-formedness)**.** *If* $\Theta'_1 \vdash \sigma_1 : \Theta_1$ *and* $\Theta'_2 \vdash \sigma_2 : \Theta_2$*, then* $\Theta'_2 \vdash \sigma_2 \ll \sigma_1 : \Theta_1$*.*

Lemma 13 (Substitution composition)**.** *If* $\Theta'_1 \vdash \sigma_1 : \Theta_1$*,* $\Theta'_2 \vdash \sigma_2 : \Theta_2$*,* $\Theta_1 \cap \Theta'_2 = \emptyset$ *and* $\Theta_1 \cap \Theta_2 = \emptyset$ *then* $\sigma_2 \circ \sigma_1 = (\sigma_2 \ll \sigma_1) \circ \sigma_2$*.*

Corollary 4 (Substitution composition commutativity)**.** *If* $\Theta'_1 \vdash \sigma_1 : \Theta_1$*,* $\Theta'_2 \vdash \sigma_2 : \Theta_2$*, and* $\Theta_1 \cap \Theta_2 = \emptyset$*,* $\Theta_1 \cap \Theta'_2 = \emptyset$*, and* $\Theta'_1 \cap \Theta_2 = \emptyset$ *then* $\sigma_2 \circ \sigma_1 = \sigma_1 \circ \sigma_2$*.*

Lemma 14 (Substitution domain weakening)**.** *If* $\Theta_2 \vdash \sigma : \Theta_1$ *then* $\Theta_2, \Theta' \vdash \sigma : \Theta_1, \Theta'$

Lemma 15 (Free variables after substitution)**.** *Suppose that* $\Theta_2 \vdash \sigma : \Theta_1$*, then*

+ *for a type* P*, the free variables of* $[\sigma]P$ *are bounded in the following way:* $\mathtt{fv}\ (P) \setminus \Theta_1 \subseteq \mathtt{fv}\ ([\sigma]P) \subseteq (\mathtt{fv}\ (P) \setminus \Theta_1) \cup \Theta_2$*;*

− *for a type* N*, the free variables of* $[\sigma]P$ *are bounded in the following way:* $\mathtt{fv}\ (N) \setminus \Theta_1 \subseteq \mathtt{fv}\ ([\sigma]N) \subseteq (\mathtt{fv}\ (N) \setminus \Theta_1) \cup \Theta_2$*.*

Lemma 16 (Free variables of a variable image)**.** *Suppose that* σ *is an arbitrary substitution, Then*

+ *if* $\alpha^{\pm} \in \mathtt{fv}\ (P)$ *then* $\mathtt{fv}\ ([\sigma]\alpha^{\pm}) \subseteq \mathtt{fv}\ ([\sigma]P)$*,*

− *if* $\alpha^{\pm} \in \mathtt{fv}\ (N)$ *then* $\mathtt{fv}\ ([\sigma]\alpha^{\pm}) \subseteq \mathtt{fv}\ ([\sigma]N)$*.*

B.1.3 Declarative Subtyping

Lemma 17 (Free Variable Propagation)**.** *In the judgments of negative subtyping or positive supertyping, free variables propagate left to right. For a context* Θ*,*

− *if* $\Theta \vdash N \leqslant M$ *then* $\mathtt{fv}\ (N) \subseteq \mathtt{fv}\ (M)$

+ *if* $\Theta \vdash P \geqslant Q$ *then* $\mathtt{fv}\ (P) \subseteq \mathtt{fv}\ (Q)$

Corollary 5 (Free Variables of mutual subtypes)**.**

− *If* $\Theta \vdash N \simeq^{\leqslant} M$ *then* $\mathtt{fv}\ N = \mathtt{fv}\ M$*,*

+ *If* $\Theta \vdash P \simeq^{\leqslant} Q$ *then* $\mathtt{fv}\ P = \mathtt{fv}\ Q$

Corollary 6. *Suppose that all the types below are well-formed in* Θ *and* $\Theta' \subseteq \Theta$*. Then*

+ $\Theta \vdash P \simeq^{\leqslant} Q$ *implies* $\Theta' \vdash P \iff \Theta' \vdash Q$

− $\Theta \vdash N \simeq^{\leqslant} M$ *implies* $\Theta' \vdash N \iff \Theta' \vdash M$

Lemma 18 (Decomposition of quantifier rules)**.** *Assuming that* $\overrightarrow{\alpha^+}$*,* $\overrightarrow{\beta^+}$*,* $\overrightarrow{\alpha^-}$*, and* $\overrightarrow{\alpha^-}$ *are disjoint from* Θ*,*

$-_R$ $\Theta \vdash N \leqslant \forall \overrightarrow{\beta^+}.\ M$ *holds if and only if* $\Theta, \overrightarrow{\beta^+} \vdash N \leqslant M$*;*

$+_R$ $\Theta \vdash P \geqslant \exists \overrightarrow{\beta^-}.\ Q$ *holds if and only if* $\Theta, \overrightarrow{\beta^-} \vdash P \geqslant Q$*;*

$-_L$ *suppose* $M \neq \forall \ldots$ *then* $\Theta \vdash \forall \overrightarrow{\alpha^+}. N \leqslant M$ *holds if and only if* $\Theta \vdash [\overrightarrow{P}/\overrightarrow{\alpha^+}]N \leqslant M$ *for some* $\Theta \vdash \overrightarrow{P}$;

$+_L$ *suppose* $Q \neq \exists \ldots$ *then* $\Theta \vdash \exists \overrightarrow{\alpha^-}. P \geqslant Q$ *holds if and only if* $\Theta \vdash [\overrightarrow{N}/\overrightarrow{\alpha^-}]P \geqslant Q$ *for some* $\Theta \vdash \overrightarrow{N}$.

Corollary 7 (Redundant quantifier elimination)**.**

$-_L$ *Suppose that* $\overrightarrow{\alpha^+} \cap \mathtt{fv}(N) = \emptyset$ *then* $\Theta \vdash \forall \overrightarrow{\alpha^+}. N \leqslant M$ *holds if and only if* $\Theta \vdash N \leqslant M$;

$-_R$ *Suppose that* $\overrightarrow{\alpha^+} \cap \mathtt{fv}(M) = \emptyset$ *then* $\Theta \vdash N \leqslant \forall \overrightarrow{\alpha^+}. M$ *holds if and only if* $\Theta \vdash N \leqslant M$;

$+_L$ *Suppose that* $\overrightarrow{\alpha^-} \cap \mathtt{fv}(P) = \emptyset$ *then* $\Theta \vdash \exists \overrightarrow{\alpha^-}. P \geqslant Q$ *holds if and only if* $\Theta \vdash P \geqslant Q$.

$+_R$ *Suppose that* $\overrightarrow{\alpha^-} \cap \mathtt{fv}(Q) = \emptyset$ *then* $\Theta \vdash P \geqslant \exists \overrightarrow{\alpha^-}. Q$ *holds if and only if* $\Theta \vdash P \geqslant Q$.

Lemma 19 (Subtypes and supertypes of a variable)**.** *Assuming* $\Theta \vdash \alpha^-$, $\Theta \vdash \alpha^+$, $\Theta \vdash N$, *and* $\Theta \vdash P$,

\+ *if* $\Theta \vdash P \geqslant \exists \overrightarrow{\alpha^-}. \alpha^+$ *or* $\Theta \vdash \exists \overrightarrow{\alpha^-}. \alpha^+ \geqslant P$ *then* $P = \exists \overrightarrow{\beta^-}. \alpha^+$ *(for some potentially empty* $\overrightarrow{\beta^-}$*)*

\- *if* $\Theta \vdash N \leqslant \forall \overrightarrow{\alpha^+}. \alpha^-$ *or* $\Theta \vdash \forall \overrightarrow{\alpha^+}. \alpha^- \leqslant N$ *then* $N = \forall \overrightarrow{\beta^+}. \alpha^-$ *(for some potentially empty* $\overrightarrow{\beta^+}$*)*

Corollary 8 (Variables have no proper subtypes and supertypes)**.** *Assuming that all mentioned types are well-formed in* Θ,

$$\Theta \vdash P \geqslant \alpha^+ \iff P = \exists \overrightarrow{\beta^-}. \alpha^+ \iff \Theta \vdash P \simeq^{\leqslant} \alpha^+ \iff P \simeq^{D} \alpha^+$$
$$\Theta \vdash \alpha^+ \geqslant P \iff P = \exists \overrightarrow{\beta^-}. \alpha^+ \iff \Theta \vdash P \simeq^{\leqslant} \alpha^+ \iff P \simeq^{D} \alpha^+$$
$$\Theta \vdash N \leqslant \alpha^- \iff N = \forall \overrightarrow{\beta^+}. \alpha^- \iff \Theta \vdash N \simeq^{\leqslant} \alpha^- \iff N \simeq^{D} \alpha^-$$
$$\Theta \vdash \alpha^- \leqslant N \iff N = \forall \overrightarrow{\beta^+}. \alpha^- \iff \Theta \vdash N \simeq^{\leqslant} \alpha^- \iff N \simeq^{D} \alpha^-$$

Lemma 20 (Subtyping context irrelevance)**.** *Suppose that all the mentioned types are well-formed in* Θ_1 *and* Θ_2. *Then*

\+ $\Theta_1 \vdash P \geqslant Q$ *is equivalent to* $\Theta_2 \vdash P \geqslant Q$;

\- $\Theta_1 \vdash N \leqslant M$ *is equivalent to* $\Theta_2 \vdash N \leqslant M$.

Lemma 21 (Weakening of subtyping context)**.** *Suppose* Θ_1 *and* Θ_2 *are contexts and* $\Theta_1 \subseteq \Theta_2$. *Then*

\+ $\Theta_1 \vdash P \geqslant Q$ *implies* $\Theta_2 \vdash P \geqslant Q$;

\- $\Theta_1 \vdash N \leqslant M$ *implies* $\Theta_2 \vdash N \leqslant M$.

Lemma 22 (Reflexivity of subtyping)**.** *Assuming all the types are well-formed in* Θ,

\- $\Theta \vdash N \leqslant N$

\+ $\Theta \vdash P \geqslant P$

Lemma 23 (Substitution preserves subtyipng)**.** *Suppose that all mentioned types are well-formed in* Θ_1, *and* σ *is a substitution* $\Theta_2 \vdash \sigma : \Theta_1$.

\- *If* $\Theta_1 \vdash N \leqslant M$ *then* $\Theta_2 \vdash [\sigma]N \leqslant [\sigma]M$.

+ *If* $\Theta_1 \vdash P \geqslant Q$ *then* $\Theta_2 \vdash [\sigma]P \geqslant [\sigma]Q$.

Corollary 9 (Substitution preserves subtyping induced equivalence)**.** *Suppose that* $\Theta \vdash \sigma : \Theta_1$. *Then*

+ *if* $\Theta_1 \vdash P$, $\Theta_1 \vdash Q$, *and* $\Theta_1 \vdash P \simeq^{\leqslant} Q$ *then* $\Theta \vdash [\sigma]P \simeq^{\leqslant} [\sigma]Q$

− *if* $\Theta_1 \vdash N$, $\Theta_1 \vdash M$, *and* $\Theta_1 \vdash N \simeq^{\leqslant} M$ *then* $\Theta \vdash [\sigma]N \simeq^{\leqslant} [\sigma]M$

Lemma 24 (Transitivity of subtyping)**.** *Assuming the types are well-formed in* Θ,

− *if* $\Theta \vdash N_1 \leqslant N_2$ *and* $\Theta \vdash N_2 \leqslant N_3$ *then* $\Theta \vdash N_1 \leqslant N_3$,

+ *if* $\Theta \vdash P_1 \geqslant P_2$ *and* $\Theta \vdash P_2 \geqslant P_3$ *then* $\Theta \vdash P_1 \geqslant P_3$.

Corollary 10 (Transitivity of equivalence)**.** *Assuming the types are well-formed in* Θ,

− *if* $\Theta \vdash N_1 \simeq^{\leqslant} N_2$ *and* $\Theta \vdash N_2 \simeq^{\leqslant} N_3$ *then* $\Theta \vdash N_1 \simeq^{\leqslant} N_3$,

+ *if* $\Theta \vdash P_1 \simeq^{\leqslant} P_2$ *and* $\Theta \vdash P_2 \simeq^{\leqslant} P_3$ *then* $\Theta \vdash P_1 \simeq^{\leqslant} P_3$.

B.1.4 Equivalence

Lemma 25 (Declarative Equivalence is invariant under bijections)**.** *Suppose* μ *is a bijection* $\mu : vars_1 \leftrightarrow vars_2$, *then*

+ $P_1 \simeq^D P_2$ *implies* $[\mu]P_1 \simeq^D [\mu]P_2$, *and there exists an inference tree of* $[\mu]P_1 \simeq^D [\mu]P_2$ *with the same shape as the one inferring* $P_1 \simeq^D P_2$;

− $N_1 \simeq^D N_2$ *implies* $[\mu]N_1 \simeq^D [\mu]N_2$, *and there exists an inference tree of* $[\mu]N_1 \simeq^D [\mu]N_2$ *with the same shape as the one inferring* $N_1 \simeq^D N_2$.

Lemma 26. *The set of free variables is invariant under equivalence.*

− *If* $N \simeq^D M$ *then* $\mathtt{fv}\ N = \mathtt{fv}\ M$ *(as sets)*

+ *If* $P \simeq^D Q$ *then* $\mathtt{fv}\ P = \mathtt{fv}\ Q$ *(as sets)*

Lemma 27 (Declarative equivalence is transitive)**.**

+ *if* $P_1 \simeq^D P_2$ *and* $P_2 \simeq^D P_3$ *then* $P_1 \simeq^D P_3$,

− *if* $N_1 \simeq^D N_2$ *and* $N_2 \simeq^D N_3$ *then* $N_1 \simeq^D N_3$.

Lemma 28 (Type well-formedness is invariant under equivalence)**.** *Mutual subtyping implies declarative equivalence.*

+ *if* $P \simeq^D Q$ *then* $\Theta \vdash P \iff \Theta \vdash Q$,

− *if* $N \simeq^D M$ *then* $\Theta \vdash N \iff \Theta \vdash M$

Lemma 29 (Soundness of equivalence)**.** *Declarative equivalence implies mutual subtyping.*

+ *if* $\Theta \vdash P$, $\Theta \vdash Q$, *and* $P \simeq^D Q$ *then* $\Theta \vdash P \simeq^{\leqslant} Q$,

− *if* $\Theta \vdash N$, $\Theta \vdash M$, *and* $N \simeq^D M$ *then* $\Theta \vdash N \simeq^{\leqslant} M$.

Lemma 30 (Subtyping induced by disjoint substitutions)**.** *Suppose that* $\Theta \vdash \sigma_1 : \Theta_1$ *and* $\Theta \vdash \sigma_2 : \Theta_1$, *where* $\Theta_i \subseteq \Theta$ *and* $\Theta_1 \cap \Theta_2 = \emptyset$. *Then*

− *assuming* $\Theta \vdash N$, $\Theta \vdash [\sigma_1]N \leqslant [\sigma_2]N$ *implies* $\Theta \vdash \sigma_i \simeq^{\leqslant} \mathsf{id} : \mathtt{fv}\ N$

+ *assuming* $\Theta \vdash P,\ \Theta \vdash [\sigma_1]P \geqslant [\sigma_2]P$ *implies* $\Theta \vdash \sigma_i \simeq^{\leqslant} \mathsf{id} : \mathtt{fv}\ P$

Corollary 11 (Substitution cannot induce proper subtypes or supertypes)**.** *Assuming all mentioned types are well-formed in* Θ *and* σ *is a substitution* $\Theta \vdash \sigma : \Theta$*,*

$$\Theta \vdash [\sigma]N \leqslant N \implies \Theta \vdash [\sigma]N \simeq^{\leqslant} N \text{ and } \Theta \vdash \sigma \simeq^{\leqslant} \mathsf{id} : \mathtt{fv}\ N$$
$$\Theta \vdash N \leqslant [\sigma]N \implies \Theta \vdash N \simeq^{\leqslant} [\sigma]N \text{ and } \Theta \vdash \sigma \simeq^{\leqslant} \mathsf{id} : \mathtt{fv}\ N$$
$$\Theta \vdash [\sigma]P \geqslant P \implies \Theta \vdash [\sigma]P \simeq^{\leqslant} P \text{ and } \Theta \vdash \sigma \simeq^{\leqslant} \mathsf{id} : \mathtt{fv}\ P$$
$$\Theta \vdash P \geqslant [\sigma]P \implies \Theta \vdash P \simeq^{\leqslant} [\sigma]P \text{ and } \Theta \vdash \sigma \simeq^{\leqslant} \mathsf{id} : \mathtt{fv}\ P$$

Lemma 31 (Mutual substitution and subtyping)**.** *Assuming that the mentioned types (*P*,* Q*,* N*, and* M*) are well-formed in* Θ *and that the substitutions (*σ_1 *and* σ_2*) have signature* $\Theta \vdash \sigma_i : \Theta$*,*

+ *if* $\Theta \vdash [\sigma_1]P \geqslant Q$ *and* $\Theta \vdash [\sigma_2]Q \geqslant P$
  then there exists a bijection $\mu : \mathtt{fv}\ P \leftrightarrow \mathtt{fv}\ Q$ *such that* $\Theta \vdash \sigma_1 \simeq^{\leqslant} \mu : \mathtt{fv}\ P$ *and* $\Theta \vdash \sigma_2 \simeq^{\leqslant} \mu^{-1} : \mathtt{fv}\ Q$*;*

– *if* $\Theta \vdash [\sigma_1]N \leqslant M$ *and* $\Theta \vdash [\sigma_2]N \leqslant M$
  then there exists a bijection $\mu : \mathtt{fv}\ N \leftrightarrow \mathtt{fv}\ M$ *such that* $\Theta \vdash \sigma_1 \simeq^{\leqslant} \mu : \mathtt{fv}\ N$ *and* $\Theta \vdash \sigma_2 \simeq^{\leqslant} \mu^{-1} : \mathtt{fv}\ M$*.*

Lemma 32 (Equivalent substitution act equivalently)**.** *Suppose that* $\Theta' \vdash \sigma_1 : \Theta$ *and* $\Theta' \vdash \sigma_2 : \Theta$ *are substitutions equivalent on their domain, that is* $\Theta' \vdash \sigma_1 \simeq^{\leqslant} \sigma_2 : \Theta$*. Then*

+ *for any* $\Theta \vdash P$*,* $\Theta' \vdash [\sigma_1]P \simeq^{\leqslant} [\sigma_2]P$*;*

– *for any* $\Theta \vdash N$*,* $\Theta' \vdash [\sigma_1]N \simeq^{\leqslant} [\sigma_2]N$*.*

Lemma 33 (Equivalence of polymorphic types)**.**

– *For* $\Theta \vdash \forall \overrightarrow{\alpha^{+}}.\ N$ *and* $\Theta \vdash \forall \overrightarrow{\beta^{+}}.\ M$*,*
  if $\Theta \vdash \forall \overrightarrow{\alpha^{+}}.\ N \simeq^{\leqslant} \forall \overrightarrow{\beta^{+}}.\ M$ *then there exists a bijection* $\mu : \overrightarrow{\beta^{+}} \cap \mathtt{fv}\ M \leftrightarrow \overrightarrow{\alpha^{+}} \cap \mathtt{fv}\ N$ *such that* $\Theta, \overrightarrow{\alpha^{+}} \vdash N \simeq^{\leqslant} [\mu]M$*,*

+ *For* $\Theta \vdash \exists \overrightarrow{\alpha^{-}}.\ P$ *and* $\Theta \vdash \exists \overrightarrow{\beta^{-}}.\ Q$*,*
  if $\Theta \vdash \exists \overrightarrow{\alpha^{-}}.\ P \simeq^{\leqslant} \exists \overrightarrow{\beta^{-}}.\ Q$ *then there exists a bijection* $\mu : \overrightarrow{\beta^{-}} \cap \mathtt{fv}\ Q \leftrightarrow \overrightarrow{\alpha^{-}} \cap \mathtt{fv}\ P$ *such that* $\Theta, \overrightarrow{\beta^{-}} \vdash P \simeq^{\leqslant} [\mu]Q$*.*

Lemma 34 (Completeness of Equivalence)**.** *Mutual subtyping implies declarative equivalence. Assuming all the types below are well-formed in* Θ*:*

+ *if* $\Theta \vdash P \simeq^{\leqslant} Q$ *then* $P \simeq^{D} Q$*,*

– *if* $\Theta \vdash N \simeq^{\leqslant} M$ *then* $N \simeq^{D} M$*.*

B.1.5 Variable Ordering

Observation 2 (Ordering is deterministic)**.** *If* `ord` *vars* `in` $N = \vec{\alpha}_1$ *and* `ord` *vars* `in` $N = \vec{\alpha}_2$ *then* $\vec{\alpha}_1 = \vec{\alpha}_2$*. If* `ord` *vars* `in` $P = \vec{\alpha}_1$ *and* `ord` *vars* `in` $P = \vec{\alpha}_2$ *then* $\vec{\alpha}_1 = \vec{\alpha}_2$*. This way, we can use* `ord` *vars* `in` N *and as a function on* N*, and* `ord` *vars* `in` P *as a function on* P*.*

Lemma 35 (Soundness of variable ordering)**.** *Variable ordering extracts used free variables.*

– $\texttt{ord}\ vars\ \texttt{in}\ N = vars \cap \texttt{fv}\ N$ *(as sets)*

+ $\texttt{ord}\ vars\ \texttt{in}\ P = vars \cap \texttt{fv}\ P$ *(as sets)*

Corollary 12 (Additivity of ordering)**.** *Variable ordering is additive (in terms of set union) with respect to its first argument.*

– $\texttt{ord}\ (vars_1 \cup vars_2)\ \texttt{in}\ N = \texttt{ord}\ vars_1\ \texttt{in}\ N \cup \texttt{ord}\ vars_2\ \texttt{in}\ N$ *(as sets)*

+ $\texttt{ord}\ (vars_1 \cup vars_2)\ \texttt{in}\ P = \texttt{ord}\ vars_1\ \texttt{in}\ P \cup \texttt{ord}\ vars_2\ \texttt{in}\ P$ *(as sets)*

Lemma 36 (Weakening of ordering)**.** *Only used variables matter in the first argument of the ordering,*

– $\texttt{ord}\ (vars \cap \texttt{fv}\ N)\ \texttt{in}\ N = \texttt{ord}\ vars\ \texttt{in}\ N$

+ $\texttt{ord}\ (vars \cap \texttt{fv}\ P)\ \texttt{in}\ P = \texttt{ord}\ vars\ \texttt{in}\ P$

Corollary 13 (Idempotency of ordering)**.**

– *If* $\texttt{ord}\ vars\ \texttt{in}\ N = \vec{\alpha}$ *then* $\texttt{ord}\ \vec{\alpha}\ \texttt{in}\ N = \vec{\alpha}$*,*

+ *If* $\texttt{ord}\ vars\ \texttt{in}\ P = \vec{\alpha}$ *then* $\texttt{ord}\ \vec{\alpha}\ \texttt{in}\ P = \vec{\alpha}$*;*

Lemma 37 (Distributivity of renaming over variable ordering)**.** *Suppose that μ is a bijection between two sets of variables $\mu : A \leftrightarrow B$.*

– *If μ is collision-free on vars and* $\texttt{fv}\ N$ *then* $[\mu](\texttt{ord}\ vars\ \texttt{in}\ N) = \texttt{ord}\ ([\mu]vars)\ \texttt{in}\ [\mu]N$

+ *If μ is collision-free on vars and* $\texttt{fv}\ P$ *then* $[\mu](\texttt{ord}\ vars\ \texttt{in}\ P) = \texttt{ord}\ ([\mu]vars)\ \texttt{in}\ [\mu]P$

Lemma 38 (Ordering is not affected by independent substitutions)**.** *Suppose that $\Theta_2 \vdash \sigma : \Theta_1$, i.e. σ maps variables from Θ_1 into types taking free variables from Θ_2, and vars is a set of variables disjoint with both Θ_1 and Θ_2, N and P are types. Then*

– $\texttt{ord}\ vars\ \texttt{in}\ [\sigma]N = \texttt{ord}\ vars\ \texttt{in}\ N$

+ $\texttt{ord}\ vars\ \texttt{in}\ [\sigma]P = \texttt{ord}\ vars\ \texttt{in}\ P$

Lemma 39 (Completeness of variable ordering)**.** *Variable ordering is invariant under equivalence. For arbitrary vars,*

– *If* $N \simeq^D M$ *then* $\texttt{ord}\ vars\ \texttt{in}\ N = \texttt{ord}\ vars\ \texttt{in}\ M$ *(as lists)*

+ *If* $P \simeq^D Q$ *then* $\texttt{ord}\ vars\ \texttt{in}\ P = \texttt{ord}\ vars\ \texttt{in}\ Q$ *(as lists)*

B.1.6 Normaliztaion

Observation 3 (Normalization is deterministic)**.** *If* $\texttt{nf}\ (N) = M$ *and* $\texttt{nf}\ (N) = M'$ *then* $M = M'$*. If* $\texttt{nf}\ (P) = Q$ *and* $\texttt{nf}\ (P) = Q'$ *then* $Q = Q'$*. This way, we can use normalization as a function.*

Lemma 40. *Free variables are not changed by the normalization*

– $\texttt{fv}\ N = \texttt{fv}\ \texttt{nf}\ (N)$

+ $\texttt{fv}\ P = \texttt{fv}\ \texttt{nf}\ (P)$

Lemma 41 (Soundness of normalization)**.**

– $N \simeq^D \mathtt{nf}\,(N)$

+ $P \simeq^D \mathtt{nf}\,(P)$

Corollary 14 (Normalization preserves well-formedness)**.**

+ $\Theta \vdash P \iff \Theta \vdash \mathtt{nf}\,(P)$,

– $\Theta \vdash N \iff \Theta \vdash \mathtt{nf}\,(N)$

Corollary 15 (Normalization preserves well-formedness of substitution)**.**
$\Theta_2 \vdash \sigma : \Theta_1 \iff \Theta_2 \vdash \mathtt{nf}\,(\sigma) : \Theta_1$

Lemma 42 (Normalization preserves substitution signature)**.** *Suppose that* σ *is a substitution,* Θ_1 *and* Θ_2 *are contexts. Then* $\Theta_2 \vdash \sigma : \Theta_1$ *implies* $\Theta_2 \vdash \mathtt{nf}\,(\sigma) : \Theta_1$.

Corollary 16 (Normalization is sound w.r.t. subtyping-induced equivalence)**.**

+ *if* $\Theta \vdash P$ *then* $\Theta \vdash P \simeq^{\leqslant} \mathtt{nf}\,(P)$,

– *if* $\Theta \vdash N$ *then* $\Theta \vdash N \simeq^{\leqslant} \mathtt{nf}\,(N)$.

Corollary 17 (Normalization preserves subtyping)**.** *Assuming all the types are well-formed in context* Θ,

+ $\Theta \vdash P \geqslant Q \iff \Theta \vdash \mathtt{nf}\,(P) \geqslant \mathtt{nf}\,(Q)$,

– $\Theta \vdash N \leqslant M \iff \Theta \vdash \mathtt{nf}\,(N) \leqslant \mathtt{nf}\,(M)$.

Corollary 18 (Normalization preserves ordering)**.** *For any vars,*

– $\mathtt{ord}\,\mathit{vars}\,\mathtt{in}\,\mathtt{nf}\,(N) = \mathtt{ord}\,\mathit{vars}\,\mathtt{in}\,M$

+ $\mathtt{ord}\,\mathit{vars}\,\mathtt{in}\,\mathtt{nf}\,(P) = \mathtt{ord}\,\mathit{vars}\,\mathtt{in}\,Q$

Lemma 43 (Distributivity of normalization over substitution)**.** *Normalization of a term distributes over substitution. Suppose that* σ *is a substitution,* N *and* P *are types. Then*

– $\mathtt{nf}\,([\sigma]N) = [\mathtt{nf}\,(\sigma)]\mathtt{nf}\,(N)$

+ $\mathtt{nf}\,([\sigma]P) = [\mathtt{nf}\,(\sigma)]\mathtt{nf}\,(P)$

where $\mathtt{nf}\,(\sigma)$ *means pointwise normalization:* $[\mathtt{nf}\,(\sigma)]\alpha^- = \mathtt{nf}\,([\sigma]\alpha^-)$.

Corollary 19 (Commutativity of normalization and renaming)**.** *Normalization of a term commutes with renaming. Suppose that* μ *is a bijection between two sets of variables* $\mu : A \leftrightarrow B$. *Then*

– $\mathtt{nf}\,([\mu]N) = [\mu]\mathtt{nf}\,(N)$

+ $\mathtt{nf}\,([\mu]P) = [\mu]\mathtt{nf}\,(P)$

Lemma 44 (Completeness of Normalization w.r.t. Declarative Equivalence)**.** *Normalization returns the same representative for equivalent types.*

– *If* $N \simeq^D M$ *then* $\mathtt{nf}\,(N) = \mathtt{nf}\,(M)$,

+ *if* $P \simeq^D Q$ *then* $\mathtt{nf}\,(P) = \mathtt{nf}\,(Q)$.

Lemma 45 (Algorithmization of Declarative Equivalence)**.** *Declarative equivalence is the equality of normal forms.*

+ $P \simeq^D Q \iff \mathtt{nf}\,(P) = \mathtt{nf}\,(Q)$,

– $N \simeq^D M \iff \mathtt{nf}(N) = \mathtt{nf}(M)$.

Corollary 20 (Completeness of Normalization w.r.t. Subtyping-Induced Equivalence)**.** *Assuming all the types below are well-formed in* Θ*:*

+ *if* $\Theta \vdash P \simeq^{\leqslant} Q$ *then* $\mathtt{nf}(P) = \mathtt{nf}(Q)$,

– *if* $\Theta \vdash N \simeq^{\leqslant} M$ *then* $\mathtt{nf}(N) = \mathtt{nf}(M)$.

Lemma 46 (Idempotence of normalization)**.** *Normalization is idempotent*

– $\mathtt{nf}(\mathtt{nf}(N)) = \mathtt{nf}(N)$

+ $\mathtt{nf}(\mathtt{nf}(P)) = \mathtt{nf}(P)$

Lemma 47. *The result of a substitution is normalized if and only if the initial type and the substitution are normalized.*

Suppose that σ *is a substitution* $\Theta_2 \vdash \sigma : \Theta_1$*,* P *is a positive type* ($\Theta_1 \vdash P$)*,* N *is a negative type* ($\Theta_1 \vdash N$)*. Then*

+ $[\sigma]P$ *is normal* $\iff \begin{cases} \sigma|_{\mathtt{fv}(P)} & \text{is normal} \\ P & \text{is normal} \end{cases}$

– $[\sigma]N$*is normal* $\iff \begin{cases} \sigma|_{\mathtt{fv}(N)} & \text{is normal} \\ N & \text{is normal} \end{cases}$

Lemma 48 (Algorithmization of subtyping-induced equivalence)**.** *Mutual subtyping is the equality of normal forms. Assuming all the types below are well-formed in* Θ*:*

+ $\Theta \vdash P \simeq^{\leqslant} Q \iff \mathtt{nf}(P) = \mathtt{nf}(Q)$,

– $\Theta \vdash N \simeq^{\leqslant} M \iff \mathtt{nf}(N) = \mathtt{nf}(M)$.

Corollary 21 (Substitution preserves declarative equivalence)**.** *Suppose that* σ *is a substitution. Then*

+ $P \simeq^D Q$ *implies* $[\sigma]P \simeq^D [\sigma]Q$

– $N \simeq^D M$ *implies* $[\sigma]N \simeq^D [\sigma]M$

B.2 Declarative Typing

Lemma 49. *If* $\Theta\,;\Gamma \vdash N_1 \bullet \vec{v} \Rrightarrow M$ *and* $\Theta \vdash N_1 \simeq^{\leqslant} N_2$ *then* $\Theta\,;\Gamma \vdash N_2 \bullet \vec{v} \Rrightarrow M$.

Lemma 50 (Declarative typing is preserved under context equivalence)**.** *Assuming* $\Theta \vdash \Gamma_1$*,* $\Theta \vdash \Gamma_2$*, and* $\Theta \vdash \Gamma_1 \simeq^{\leqslant} \Gamma_2$*:*

+ *for any tree* T_1 *inferring* $\Theta; \Gamma_1 \vdash v : P$*, there exists a tree* T_2 *inferring* $\Theta; \Gamma_2 \vdash v : P$.

– *for any tree* T_1 *inferring* $\Theta; \Gamma_1 \vdash c : N$*, there exists a tree* T_2 *inferring* $\Theta; \Gamma_2 \vdash c : N$.

• *for any tree* T_1 *inferring* $\Theta\,;\Gamma_1 \vdash N \bullet \vec{v} \Rrightarrow M$*, there exists a tree* T_2 *inferring* $\Theta\,;\Gamma_2 \vdash N \bullet \vec{v} \Rrightarrow M$.

B.3 Relation to System F

Lemma 51 (Subtyping elaboration term can be removed)**.**

- *For any* Θ, N, *and* M, $\Theta \vdash N \leqslant M$ *holds if and only if there exists* t *such that* $\Theta \vdash N \leqslant M \rightsquigarrow t$;
- + *For any* Θ, P, *and* Q, $\Theta \vdash P \geqslant Q$ *holds if and only if there exists* t *such that* $\Theta \vdash P \geqslant Q \rightsquigarrow t$.

Observation 4 (Type depolarization distributes over substitution)**.**

- + $|[\sigma]N| = [|\sigma|]|N|$,
- − $|[\sigma]P| = [|\sigma|]|P|$.

Lemma 52 (Soundness of Subtyping Elaboration)**.**

- − *If* $\Theta \vdash N \leqslant M \rightsquigarrow t$ *then* $|\Theta|; \cdot \vdash t : |N| \to |M|$;
- + *if* $\Theta \vdash P \geqslant Q \rightsquigarrow t$ *then* $|\Theta|; \cdot \vdash t : |Q| \to |P|$.

Lemma 53 (Soundness of $\mathrm{F}^{\pm}_{\exists}$ w.r.t. System F)**.** *A judgment inferred by* $F^{\pm}_{\exists}$ *is derivable in System F.*

- + *If* $\Theta; \Gamma \vdash v : P \rightsquigarrow t$ *then* $|\Theta|; |\Gamma| \vdash t : |P|$;
- − *if* $\Theta; \Gamma \vdash c : N \rightsquigarrow t$ *then* $|\Theta|; |\Gamma| \vdash t : |N|$;
- • *if* $\Theta; \Gamma \vdash N \bullet \vec{v} \Rrightarrow M \rightsquigarrow e; \vec{t}$ *then* $|\Theta|; |\Gamma|, x : |N| \vdash e\ (x\,\vec{t}) : |M|$.

Lemma 54 (Polarization commutes with substitution)**.** $\downarrow[A/\alpha]T\downarrow = [\downarrow A\downarrow/\alpha^{+}]\downarrow T\downarrow$

Observation 5. *For any* Θ, Γ, t, *and* T, *there exists* c *such that* $\Theta; \Gamma \vdash t : T \rightsquigarrow^{\pm} c$ *if and only if* $\Theta; \Gamma \vdash t : T$.

Lemma 55 (Type polarization agrees with well-formedness)**.** *If* $\Theta \vdash T$ *then* $\downarrow\Theta\downarrow \vdash \downarrow T\downarrow$.

Lemma 56 (Polarization preserves typing)**.** *If* $\Theta; \Gamma \vdash t : T \rightsquigarrow^{\pm} c$ *then* $\downarrow\Theta\downarrow; \downarrow\Gamma\downarrow \vdash c : \uparrow\downarrow T\downarrow$.

B.4 Theorem Statements: Algorithmic

B.4.1 Algorithmic Type Well-formedness

Lemma 57. *If* $\Theta\,; \Gamma \vdash N_1 \bullet \vec{v} \Rrightarrow M$ *and* $\Theta \vdash N_1 \simeq^{\leqslant} N_2$ *then* $\Theta\,; \Gamma \vdash N_2 \bullet \vec{v} \Rrightarrow M$.

Lemma 58 (Soundness of algorithmic type well-formedness)**.**

- + *if* $\Theta\,; \widehat{\Theta} \vdash P$ *then* $\mathtt{fv}\,(P) \subseteq \Theta$ *and* $\mathtt{fav}(P) \subseteq \widehat{\Theta}$;
- − *if* $\Theta\,; \widehat{\Theta} \vdash N$ *then* $\mathtt{fv}\,(N) \subseteq \Theta$ *and* $\mathtt{fav}(N) \subseteq \widehat{\Theta}$.

Lemma 59 (Completeness of algorithmic type well-formedness)**.** *In the well-formedness judgment, only used variables matter:*

- + *if* $\Theta_1 \cap \mathtt{fv}\,P = \Theta_2 \cap \mathtt{fv}\,P$ *and* $\widehat{\Theta}_1 \cap \mathtt{fav}P = \widehat{\Theta}_2 \cap \mathtt{fav}P$ *then* $\Theta_1\,; \widehat{\Theta}_1 \vdash P \iff \Theta_2\,; \widehat{\Theta}_2 \vdash P$, *and*
- − *if* $\Theta_1 \cap \mathtt{fv}\,N = \Theta_2 \cap \mathtt{fv}\,N$ *and* $\widehat{\Theta}_1 \cap \mathtt{fav}N = \widehat{\Theta}_2 \cap \mathtt{fav}N$ *then* $\Theta_1\,; \widehat{\Theta}_1 \vdash N \iff \Theta_2\,; \widehat{\Theta}_2 \vdash N$.

Lemma 60 (Variable algorithmization agrees with well-formedness)**.**

+ $\Theta, \overrightarrow{\alpha} \vdash P$ *implies* $\Theta ; \overrightarrow{\widehat{\alpha}} \vdash [\overrightarrow{\widehat{\alpha}}/\overrightarrow{\alpha}]P$*;*
− $\Theta, \overrightarrow{\alpha} \vdash N$ *implies* $\Theta ; \overrightarrow{\widehat{\alpha}} \vdash [\overrightarrow{\widehat{\alpha}}/\overrightarrow{\alpha}]N$*.*

Lemma 61 (Variable de-algorithmization agrees with well-formedness)**.**

+ $\Theta ; \overrightarrow{\widehat{\alpha}} \vdash P$ *implies* $\Theta, \overrightarrow{\alpha} \vdash [\overrightarrow{\alpha}/\overrightarrow{\widehat{\alpha}}]P$*;*
− $\Theta ; \overrightarrow{\widehat{\alpha}} \vdash N$ *implies* $\Theta, \overrightarrow{\alpha} \vdash [\overrightarrow{\alpha}/\overrightarrow{\widehat{\alpha}}]N$*.*

Corollary 22 (Well-formedness Algorithmic Context Weakening)**.** *Suppose that* $\Theta_1 \subseteq \Theta_2$*, and* $\widehat{\Theta}_1 \subseteq \widehat{\Theta}_2$*. Then*

+ *if* $\Theta_1 ; \widehat{\Theta}_1 \vdash P$ *implies* $\Theta_2 ; \widehat{\Theta}_2 \vdash P$*,*
− *if* $\Theta_1 ; \widehat{\Theta}_1 \vdash N$ *implies* $\Theta_2 ; \widehat{\Theta}_2 \vdash N$*.*

B.4.2 Algorithmic Substitution

Lemma 62 (Determinacy of typing algorithm)**.** *Suppose that* $\Theta \vdash \Gamma$ *and* $\Theta \vdash^{\supseteq} \Xi$*. Then*

+ *If* $\Theta ; \Gamma \vDash v : P$ *and* $\Theta ; \Gamma \vDash v : P'$ *then* $P = P'$*.*
− *If* $\Theta ; \Gamma \vDash c : N$ *and* $\Theta ; \Gamma \vDash c : N'$ *then* $N = N'$*.*
• *If* $\Theta ; \Gamma ; \Xi \vDash N \bullet \vec{v} \Longrightarrow M \dashv \Xi' ; C$ *and* $\Theta ; \Gamma ; \Xi \vDash N \bullet \vec{v} \Longrightarrow M' \dashv \Xi' ; C'$ *then* $M = M'$*,* $\Xi = \Xi'$*, and* $C = C'$*.*

Lemma 63 (Algorithmic Substitution Strengthening)**.** *Restricting the substitution to the algorithmic variables of the substitution subject does not affect the result. Suppose that* $\widehat{\sigma}$ *is an algorithmic substitution,* P *and* N *are algorithmic types. Then*

+ $[\widehat{\sigma}]P = [\widehat{\sigma}|_{\mathtt{fav}P}]P$*,*
− $[\widehat{\sigma}]N = [\widehat{\sigma}|_{\mathtt{fav}N}]N$

Lemma 64 (Substitutions equal on the algorithmic variables)**.** *Suppose that* $\widehat{\sigma}_1$ *and* $\widehat{\sigma}_2$ *are normalized substitutions of signature* $\Xi \vdash \widehat{\sigma}_i : \widehat{\Theta}$*. Then*

+ *for a normalized type* $\Theta ; \widehat{\Theta} \vdash P$*, if* $[\widehat{\sigma}_1]P = [\widehat{\sigma}_2]P$ *then* $\widehat{\sigma}_1|_{(\mathtt{fav}P)} = \widehat{\sigma}_2|_{(\mathtt{fav}P)}$*;*
− *for a normalized type* $\Theta ; \widehat{\Theta} \vdash N$*, if* $[\widehat{\sigma}_1]N = [\widehat{\sigma}_2]N$ *then* $\widehat{\sigma}_1|_{(\mathtt{fav}N)} = \widehat{\sigma}_2|_{(\mathtt{fav}N)}$*.*

Corollary 23 (Substitutions equivalent on the algorithmic variables)**.** *Suppose that* $\widehat{\sigma}_1$ *and* $\widehat{\sigma}_2$ *are substitutions of signature* $\Xi \vdash \widehat{\sigma}_i : \widehat{\Theta}$ *where* $\Theta \vdash^{\supseteq} \Xi$*. Then*

+ *for a type* $\Theta ; \widehat{\Theta} \vdash P$*, if* $\Theta \vdash [\widehat{\sigma}_1]P \simeq^{\leqslant} [\widehat{\sigma}_2]P$ *then* $\Xi \vdash \widehat{\sigma}_1 \simeq^{\leqslant} \widehat{\sigma}_2 : \mathtt{fav}P$*;*
− *for a type* $\Theta ; \widehat{\Theta} \vdash N$*, if* $\Theta \vdash [\widehat{\sigma}_1]N \simeq^{\leqslant} [\widehat{\sigma}_2]N$ *then* $\Xi \vdash \widehat{\sigma}_1 \simeq^{\leqslant} \widehat{\sigma}_2 : \mathtt{fav}N$*.*

B.4.3 Algorithmic Normalization

Lemma 65 (Determinacy of typing algorithm)**.** *Suppose that* $\Theta \vdash \Gamma$ *and* $\Theta \vdash^{\supseteq} \Xi$*. Then*

+ *If* $\Theta ; \Gamma \vDash v : P$ *and* $\Theta ; \Gamma \vDash v : P'$ *then* $P = P'$*.*
− *If* $\Theta ; \Gamma \vDash c : N$ *and* $\Theta ; \Gamma \vDash c : N'$ *then* $N = N'$*.*

- *If* $\Theta\,;\Gamma\,;\Xi \vdash N \bullet \vec{v} \Longrightarrow M \dashv \Xi'\,; C$ *and* $\Theta\,;\Gamma\,;\Xi \vdash N \bullet \vec{v} \Longrightarrow M' \dashv \Xi'\,; C'$ *then* $M = M'$, $\Xi = \Xi'$, *and* $C = C'$.

Lemma 66 (Algorithmic variables are not changed by the normalization)**.**

- $\mathtt{fav} N \equiv \mathtt{favnf}\,(N)$
+ $\mathtt{fav} P \equiv \mathtt{favnf}\,(P)$

Lemma 67 (Soundness of normalization of algorithmic types)**.**

- $N \simeq^{D} \mathtt{nf}\,(N)$
+ $P \simeq^{D} \mathtt{nf}\,(P)$

B.4.4 Algorithmic Equivalence

Lemma 68 (Algorithmic type well-formedness is invariant under equivalence)**.** *Mutual subtyping implies declarative equivalence.*

+ *if* $P \simeq^{D} Q$ *then* $\Theta\,;\widehat{\Theta} \vdash P \iff \Theta\,;\widehat{\Theta} \vdash Q$,
- *if* $N \simeq^{D} M$ *then* $\Theta\,;\widehat{\Theta} \vdash N \iff \Theta\,;\widehat{\Theta} \vdash M$

Corollary 24 (Normalization preserves well-formedness of algorithmic types)**.**

+ $\Theta\,;\widehat{\Theta} \vdash P \iff \Theta\,;\widehat{\Theta} \vdash \mathtt{nf}\,(P)$,
- $\Theta\,;\widehat{\Theta} \vdash N \iff \Theta\,;\widehat{\Theta} \vdash \mathtt{nf}\,(N)$

Corollary 25 (Normalization preserves the signature of the algorithmic substitution)**.** $\Xi \vdash \widehat{\sigma} : \widehat{\Theta} \iff \Xi \vdash \mathtt{nf}\,(\widehat{\sigma}) : \widehat{\Theta}$, $\Theta \vdash \widehat{\sigma} : \widehat{\Theta} \iff \Theta \vdash \mathtt{nf}\,(\widehat{\sigma}) : \widehat{\Theta}$.

Corollary 26 (Algorithmic substitution equivalence becomes equality after normalization)**.** *Suppose that* $\Xi \vdash \widehat{\sigma}_1 : \widehat{\Theta}'$ *and* $\Xi \vdash \widehat{\sigma}_2 : \widehat{\Theta}'$ *are algorithmic substitutions and* $\widehat{\Theta} \subseteq \widehat{\Theta}'$. *Then* $\Xi \vdash \widehat{\sigma}_1 \simeq^{\leqslant} \widehat{\sigma}_2 : \widehat{\Theta} \iff \mathtt{nf}\,(\widehat{\sigma}_1)|_{\widehat{\Theta}} = \mathtt{nf}\,(\widehat{\sigma}_2)|_{\widehat{\Theta}}$.

B.4.5 Unification Constraint Merge

Observation 6 (Unification Constraint Merge Determinism)**.** *Suppose that* $\Xi \vdash UC_1$ *and* $\Xi \vdash UC_2$ *If* $\Xi \vdash UC_1 \,\&\, UC_2 = UC$ *and* $\Xi \vdash UC_1 \,\&\, UC_2 = UC'$ *are defined then* $UC = UC'$.

Lemma 69 (Soundness of Unification Constraint Merge)**.** *Suppose that* $\Xi \vdash UC_1$ *and* $\Xi \vdash UC_2$ *are normalized unification constraints. If* $\Xi \vdash UC_1 \,\&\, UC_2 = UC$ *is defined then* $UC = UC_1 \cup UC_2$.

Corollary 27. *Suppose that* $\Xi \vdash UC_1$ *and* $\Xi \vdash UC_2$ *are normalized unification constraints. If* $\Xi \vdash UC_1 \,\&\, UC_2 = UC$ *is defined then*

1. $\Xi \vdash UC$ *is normalized unification constraint,*
2. *for any substitution* $\Xi \vdash \widehat{\sigma} : \mathtt{dom}\,(UC)$, $\Xi \vdash \widehat{\sigma} : UC$ *implies* $\Xi \vdash \widehat{\sigma} : UC_1$ *and* $\Xi \vdash \widehat{\sigma} : UC_2$.

Lemma 70 (Completeness of Unification Constraint Entry Merge)**.** *For a fixed context* Θ, *suppose that* $\Theta \vdash ue_1$ *and* $\Theta \vdash ue_2$ *are matching constraint entries.*

+ *for a type* P *such that* $\Theta \vdash P : ue_1$ *and* $\Theta \vdash P : ue_2$, $\Theta \vdash ue_1 \,\&\, ue_2 = ue$ *is defined and* $\Theta \vdash P : ue$.

− *for a type* N *such that* $\Theta \vdash N : ue_1$ *and* $\Theta \vdash N : ue_2$, $\Theta \vdash ue_1 \,\&\, ue_2 = ue$ *is defined and* $\Theta \vdash N : ue$.

Lemma 71 (Completeness of Unification Constraint Merge)**.** *Suppose that* $\Xi \vdash UC_1$ *and* $\Xi \vdash UC_2$. *Then for any* $\widehat{\Theta} \supseteq \mathtt{dom}(UC_1) \cup \mathtt{dom}(UC_2)$ *and substitution* $\Xi \vdash \widehat{\sigma} : \widehat{\Theta}$ *such that* $\Xi \vdash \widehat{\sigma} : UC_1$ *and* $\Xi \vdash \widehat{\sigma} : UC_2$,

1. $\Xi \vdash UC_1 \,\&\, UC_2 = UC$ *is defined and*
2. $\Xi \vdash \widehat{\sigma} : UC$.

B.4.6 Unification

Observation 7 (Unification Determinism)**.**

+ *If* $\Theta; \Xi \vDash P \overset{u}{\simeq} Q \dashv UC$ *and* $\Theta; \Xi \vDash P \overset{u}{\simeq} Q \dashv UC'$ *then* $UC = UC'$.

− *If* $\Theta; \Xi \vDash N \overset{u}{\simeq} M \dashv UC$ *and* $\Theta; \Xi \vDash N \overset{u}{\simeq} M \dashv UC'$ *then* $UC = UC'$.

Lemma 72 (Soundness of Unification)**.**

+ *For normalized* P *and* Q *such that* $\Theta\,; \mathtt{dom}(\Xi) \vdash P$ *and* $\Theta \vdash Q$, *if* $\Theta; \Xi \vDash P \overset{u}{\simeq} Q \dashv UC$ *then* $\Xi \vdash UC : \mathtt{fav} P$ *and for any normalized* $\widehat{\sigma}$ *such that* $\Xi \vdash \widehat{\sigma} : UC$, $[\widehat{\sigma}]P = Q$.

− *For normalized* N *and* M *such that* $\Theta\,; \mathtt{dom}(\Xi) \vdash N$ *and* $\Theta \vdash M$, *if* $\Theta; \Xi \vDash N \overset{u}{\simeq} M \dashv UC$ *then* $\Xi \vdash UC : \mathtt{fav} N$ *and for any normalized* $\widehat{\sigma}$ *such that* $\Xi \vdash \widehat{\sigma} : UC$, $[\widehat{\sigma}]N = M$.

Lemma 73 (Completeness of Unification)**.**

+ *For normalized* P *and* Q *such that* $\Theta\,; \mathtt{dom}(\Xi) \vdash P$ *and* $\Theta \vdash Q$, *suppose that there exists* $\Xi \vdash \widehat{\sigma} : \mathtt{fav}(P)$ *such that* $[\widehat{\sigma}]P = Q$, *then* $\Theta; \Xi \vDash P \overset{u}{\simeq} Q \dashv UC$ *for some* UC.

− *For normalized* N *and* M *such that* $\Theta\,; \mathtt{dom}(\Xi) \vdash N$ *and* $\Theta \vdash M$, *suppose that there exists* $\Xi \vdash \widehat{\sigma} : \mathtt{fav}(N)$ *such that* $[\widehat{\sigma}]N = M$, *then* $\Theta; \Xi \vDash N \overset{u}{\simeq} M \dashv UC$ *for some* UC.

B.4.7 Anti-unification

Observation 8 (Determinism of Anti-unification Algorithm)**.**

+ *If* $\Theta \vDash P_1 \overset{a}{\simeq} P_2 \dashv (\widehat{\Theta}, Q, \widehat{\tau}_1, \widehat{\tau}_2)$ *and* $\Theta \vDash P_1 \overset{a}{\simeq} P_2 \dashv (\widehat{\Theta}', Q', \widehat{\tau}_1', \widehat{\tau}_2')$, *then* $\widehat{\Theta} = \widehat{\Theta}'$, $Q = Q'$, $\widehat{\tau}_1 = \widehat{\tau}_1'$, *and* $\widehat{\tau}_2 = \widehat{\tau}_2'$.

− *If* $\Theta \vDash N_1 \overset{a}{\simeq} N_2 \dashv (\widehat{\Theta}, M, \widehat{\tau}_1, \widehat{\tau}_2)$ *and* $\Theta \vDash N_1 \overset{a}{\simeq} N_2 \dashv (\widehat{\Theta}', M', \widehat{\tau}_1', \widehat{\tau}_2')$, *then* $\widehat{\Theta} = \widehat{\Theta}'$, $M = M'$, $\widehat{\tau}_1 = \widehat{\tau}_1'$, *and* $\widehat{\tau}_2 = \widehat{\tau}_2'$.

Observation 9 (Uniqueness of Anti-unification Variable Names)**.** *Names of the anti-unification variables are uniquely defined by the types they are mapped to by the resulting substitutions.*

+ *Assuming* P_1 *and* P_2 *are normalized, if* $\Theta \vDash P_1 \overset{a}{\simeq} P_2 \dashv (\widehat{\Theta}, Q, \widehat{\tau}_1, \widehat{\tau}_2)$ *then for any* $\widehat{\beta}^- \in \widehat{\Theta}$, $\widehat{\beta}^- = \widehat{\alpha}^-_{\{[\widehat{\tau}_1]\widehat{\beta}^-, [\widehat{\tau}_2]\widehat{\beta}^-\}}$

– *Assuming* N_1 *and* N_2 *are normalized, if* $\Theta \vDash N_1 \overset{a}{\simeq} N_2 \dashv (\widehat{\Theta}, M, \widehat{\tau}_1, \widehat{\tau}_2)$ *then for any* $\widehat{\beta}^- \in \widehat{\Theta}$, $\widehat{\beta}^- = \widehat{\alpha}^-_{\{[\widehat{\tau}_1]\widehat{\beta}^-, [\widehat{\tau}_2]\widehat{\beta}^-\}}$

Lemma 74 (Soundness of Anti-Unification)**.**

+ *Assuming* P_1 *and* P_2 *are normalized, if* $\Theta \vDash P_1 \overset{a}{\simeq} P_2 \dashv (\widehat{\Theta}, Q, \widehat{\tau}_1, \widehat{\tau}_2)$ *then*

1. $\Theta ; \widehat{\Theta} \vdash Q$,
2. $\Theta ; \cdot \vdash \widehat{\tau}_i : \widehat{\Theta}$ *for* $i \in \{1, 2\}$ *are anti-unification substitutions, and*
3. $[\widehat{\tau}_i] Q = P_i$ *for* $i \in \{1, 2\}$.

– *Assuming* N_1 *and* N_2 *are normalized, if* $\Theta \vDash N_1 \overset{a}{\simeq} N_2 \dashv (\widehat{\Theta}, M, \widehat{\tau}_1, \widehat{\tau}_2)$ *then*

1. $\Theta ; \widehat{\Theta} \vdash M$,
2. $\Theta ; \cdot \vdash \widehat{\tau}_i : \widehat{\Theta}$ *for* $i \in \{1, 2\}$ *are anti-unification substitutions, and*
3. $[\widehat{\tau}_i] M = N_i$ *for* $i \in \{1, 2\}$.

Lemma 75 (Completeness of Anti-Unification)**.**

+ *Assume that* P_1 *and* P_2 *are normalized, and there exists* $(\widehat{\Theta}', Q', \widehat{\tau}'_1, \widehat{\tau}'_2)$ *such that*

1. $\Theta ; \widehat{\Theta}' \vdash Q'$,
2. $\Theta ; \cdot \vdash \widehat{\tau}'_i : \widehat{\Theta}'$ *for* $i \in \{1, 2\}$ *are anti-unification substitutions, and*
3. $[\widehat{\tau}'_i] Q' = P_i$ *for* $i \in \{1, 2\}$.

Then the anti-unification algorithm terminates, that is there exists $(\widehat{\Theta}, Q, \widehat{\tau}_1, \widehat{\tau}_2)$ *such that* $\Theta \vDash P_1 \overset{a}{\simeq} P_2 \dashv (\widehat{\Theta}, Q, \widehat{\tau}_1, \widehat{\tau}_2)$

– *Assume that* N_1 *and* N_2 *are normalized, and there exists* $(\widehat{\Theta}', M', \widehat{\tau}'_1, \widehat{\tau}'_2)$ *such that*

1. $\Theta ; \widehat{\Theta}' \vdash M'$,
2. $\Theta ; \cdot \vdash \widehat{\tau}'_i : \widehat{\Theta}'$ *for* $i \in \{1, 2\}$, *are anti-unification substitutions, and*
3. $[\widehat{\tau}'_i] M' = N_i$ *for* $i \in \{1, 2\}$.

Then the anti-unification algorithm succeeds, that is there exists $(\widehat{\Theta}, M, \widehat{\tau}_1, \widehat{\tau}_2)$ *such that* $\Theta \vDash N_1 \overset{a}{\simeq} N_2 \dashv (\widehat{\Theta}, M, \widehat{\tau}_1, \widehat{\tau}_2)$.

Lemma 76 (Initiality of Anti-Unification)**.**

+ *Assume that* P_1 *and* P_2 *are normalized, and* $\Theta \vDash P_1 \overset{a}{\simeq} P_2 \dashv (\widehat{\Theta}, Q, \widehat{\tau}_1, \widehat{\tau}_2)$, *then* $(\widehat{\Theta}, Q, \widehat{\tau}_1, \widehat{\tau}_2)$ *is more specific than any other sound anti-unifier* $(\widehat{\Theta}', Q', \widehat{\tau}'_1, \widehat{\tau}'_2)$, *i.e. if*

1. $\Theta ; \widehat{\Theta}' \vdash Q'$,
2. $\Theta ; \cdot \vdash \widehat{\tau}'_i : \widehat{\Theta}'$ *for* $i \in \{1, 2\}$, *and*
3. $[\widehat{\tau}'_i] Q' = P_i$ *for* $i \in \{1, 2\}$

then there exists $\widehat{\rho}$ *such that* $\Theta ; \widehat{\Theta} \vdash \widehat{\rho} : (\widehat{\Theta}'|_{\mathtt{fav}\, Q'})$ *and* $[\widehat{\rho}]\, Q' = Q$. *Moreover,* $[\widehat{\rho}]\widehat{\beta}^{-}$ *can be uniquely determined by* $[\widehat{\tau}_1']\widehat{\beta}^{-}$, $[\widehat{\tau}_2']\widehat{\beta}^{-}$, *and* Θ.

– *Assume that* N_1 *and* N_2 *are normalized, and* $\Theta \vDash N_1 \overset{a}{\simeq} N_2 \dashv (\widehat{\Theta}, M, \widehat{\tau}_1, \widehat{\tau}_2)$, *then* $(\widehat{\Theta}, M, \widehat{\tau}_1, \widehat{\tau}_2)$ *is more specific than any other sound anti-unifier* $(\widehat{\Theta}', M', \widehat{\tau}_1', \widehat{\tau}_2')$, *i.e. if*

1. $\Theta ; \widehat{\Theta}' \vdash M'$,
2. $\Theta ; \cdot \vdash \widehat{\tau}_i' : \widehat{\Theta}'$ *for* $i \in \{1, 2\}$, *and*
3. $[\widehat{\tau}_i']\, M' = N_i$ *for* $i \in \{1, 2\}$

then there exists $\widehat{\rho}$ *such that* $\Theta ; \widehat{\Theta} \vdash \widehat{\rho} : (\widehat{\Theta}'|_{\mathtt{fav}\, M'})$ *and* $[\widehat{\rho}]\, M' = M$. *Moreover,* $[\widehat{\rho}]\widehat{\beta}^{-}$ *can be uniquely determined by* $[\widehat{\tau}_1']\widehat{\beta}^{-}$, $[\widehat{\tau}_2']\widehat{\beta}^{-}$, *and* Θ.

B.4.8 Upper Bounds

Observation 10 (Determinism of Least Upper Bound algorithm)**.** *For types* $\Theta \vdash P_1$, *and* $\Theta \vdash P_2$, *if* $\Theta \vDash P_1 \vee P_2 = Q$ *and* $\Theta \vDash P_1 \vee P_2 = Q'$ *then* $Q = Q'$.

Lemma 77 (Characterization of the Supertypes)**.** *Let us define the set of upper bounds of a positive type* $\mathsf{UB}(P)$ *in the following way:*

$$\frac{}{\Theta \vdash P \qquad \mathsf{UB}(\Theta \vdash P)}$$

$$\frac{}{\Theta \vdash \beta^{+} \qquad \{\exists \overrightarrow{\alpha^{-}}.\, \beta^{+} \mid \textit{for } \overrightarrow{\alpha^{-}}\}}$$

$$\frac{}{\Theta \vdash \exists \overrightarrow{\beta^{-}}.\, Q \qquad \mathsf{UB}(\Theta, \overrightarrow{\beta^{-}} \vdash Q) \textit{ not using } \overrightarrow{\beta^{-}}}$$

$$\frac{}{\Theta \vdash \downarrow M \qquad \left\{ \exists \overrightarrow{\alpha^{-}}.\, \downarrow M' \;\middle|\; \begin{array}{l} \textit{for } \overrightarrow{\alpha^{-}}, M', \textit{and } \vec{N} \textit{ s.t.} \\ \Theta \vdash N_i, \Theta, \overrightarrow{\alpha^{-}} \vdash M', \textit{and } [\vec{N}/\overrightarrow{\alpha^{-}}]\downarrow M' \simeq^{D} \downarrow M \end{array} \right\}}$$

Then $\mathsf{UB}(\Theta \vdash P) \equiv \{Q \mid \Theta \vdash Q \geqslant P\}$.

Lemma 78 (Characterization of the Normalized Supertypes)**.** *For a normalized positive type* $P = \mathtt{nf}\,(P)$, *let us define the set of normalized upper bounds in the following way:*

$$\frac{}{\Theta \vdash P \qquad \mathsf{NFUB}(\Theta \vdash P)}$$

$$\frac{}{\Theta \vdash \beta^{+} \qquad \{\beta^{+}\}}$$

$$\frac{}{\Theta \vdash \exists \overrightarrow{\beta^{-}}.\, P \qquad \mathsf{NFUB}(\Theta, \overrightarrow{\beta^{-}} \vdash P) \textit{ not using } \overrightarrow{\beta^{-}}}$$

$$\frac{}{\Theta \vdash \downarrow M \qquad \left\{ \exists \overrightarrow{\alpha^{-}}.\, \downarrow M' \;\middle|\; \begin{array}{l} \textit{for } \overrightarrow{\alpha^{-}}, M', \textit{and } \vec{N} \textit{ s.t. } \mathtt{ord}\, \overrightarrow{\alpha^{-}}\, \mathtt{in}\, M' = \overrightarrow{\alpha^{-}}, \\ \Theta \vdash N_i, \Theta, \overrightarrow{\alpha^{-}} \vdash M', \textit{and } [\vec{N}/\overrightarrow{\alpha^{-}}]\downarrow M' = \downarrow M \end{array} \right\}}$$

Then $\mathsf{NFUB}(\Theta \vdash P) \equiv \{\mathtt{nf}\,(Q) \mid \Theta \vdash Q \geqslant P\}$.

Lemma 79. *Upper bounds of a type do not depend on the context as soon as the type is well-formed in it.*

If $\Theta_1 \vdash P$ *and* $\Theta_2 \vdash P$ *then* $\mathsf{UB}(\Theta_1 \vdash P) = \mathsf{UB}(\Theta_2 \vdash P)$ *and* $\mathsf{NFUB}(\Theta_1 \vdash P) = \mathsf{NFUB}(\Theta_2 \vdash P)$

Lemma 80 (Soundness of the Least Upper Bound). *For types* $\Theta \vdash P_1$, *and* $\Theta \vdash P_2$, *if* $\Theta \vDash P_1 \vee P_2 = Q$ *then*

(i) $\Theta \vdash Q$

(ii) $\Theta \vdash Q \geqslant P_1$ *and* $\Theta \vdash Q \geqslant P_2$

Lemma 81 (Completeness and Initiality of the Least Upper Bound). *For types* $\Theta \vdash P_1$, $\Theta \vdash P_2$, *and* $\Theta \vdash Q$ *such that* $\Theta \vdash Q \geqslant P_1$ *and* $\Theta \vdash Q \geqslant P_2$, *there exists* Q' *s.t.* $\Theta \vDash P_1 \vee P_2 = Q'$ *and* $\Theta \vdash Q \geqslant Q'$.

B.4.9 Upgrade

Observation 11 (Upgrade determinism). *Assuming* P *is well-formed in* $\Theta \subseteq \Theta_0$, *if* $\texttt{upgrade}\ \Theta \vdash P\ \texttt{to}\ \Theta_0 = Q$ *and* $\texttt{upgrade}\ \Theta \vdash P\ \texttt{to}\ \Theta_0 = Q'$ *are defined then* $Q = Q'$.

Lemma 82 (Soundness of Upgrade). *Assuming* P *is well-formed in* $\Theta = \Theta_0, \overrightarrow{\alpha^{\pm}}$, *if* $\texttt{upgrade}\ \Theta \vdash P\ \texttt{to}\ \Theta_0 = Q$ *then*

1. $\Theta_0 \vdash Q$
2. $\Theta \vdash Q \geqslant P$

Lemma 83 (Completeness and Initiality of Upgrade). *The upgrade returns the least* Θ*-supertype of* P *well-formed in* Θ_0. *Assuming* P *is well-formed in* $\Theta = \Theta_0, \overrightarrow{\alpha^{\pm}}$, *For any* Q' *such that*

1. $\Theta_0 \vdash Q'$ *and*
2. $\Theta \vdash Q' \geqslant P$,

the result of the upgrade algorithm Q *exists (*$\texttt{upgrade}\ \Theta \vdash P\ \texttt{to}\ \Theta_0 = Q$*) and satisfies* $\Theta_0 \vdash Q' \geqslant Q$.

B.4.10 Constraint Satisfaction

Lemma 84 (Any constraint is satisfiable). *Suppose that* $\Xi \vdash C$ *and* $\widehat{\Theta}$ *is a set such that* $\texttt{dom}\,(C) \subseteq \widehat{\Theta} \subseteq \texttt{dom}\,(\Xi)$. *Then there exists* $\widehat{\sigma}$ *such that* $\Xi \vdash \widehat{\sigma} : \widehat{\Theta}$ *and* $\Xi \vdash \widehat{\sigma} : C$.

Lemma 85 (Constraint Entry Satisfaction is Stable under Equivalence).

– *If* $\Theta \vdash N_1 : e$ *and* $\Theta \vdash N_1 \simeq^{\leqslant} N_2$ *then* $\Theta \vdash N_2 : e$.

\+ *If* $\Theta \vdash P_1 : e$ *and* $\Theta \vdash P_1 \simeq^{\leqslant} P_2$ *then* $\Theta \vdash P_2 : e$.

Corollary 28 (Constraint Satisfaction is stable under Equivalence).
If $\Xi \vdash \widehat{\sigma}_1 : C$ *and* $\Xi \vdash \widehat{\sigma}_1 \simeq^{\leqslant} \widehat{\sigma}_2 : \texttt{dom}\,(C)$ *then* $\Xi \vdash \widehat{\sigma}_2 : C$;
if $\Xi \vdash \widehat{\sigma}_1 : UC$ *and* $\Xi \vdash \widehat{\sigma}_1 \simeq^{\leqslant} \widehat{\sigma}_2 : \texttt{dom}\,(C)$ *then* $\Xi \vdash \widehat{\sigma}_2 : UC$.

Corollary 29 (Normalization preserves Constraint Satisfaction).
If $\Xi \vdash \widehat{\sigma} : C$ *then* $\Xi \vdash \texttt{nf}\,(\widehat{\sigma}) : C$;
if $\Xi \vdash \widehat{\sigma} : UC$ *then* $\Xi \vdash \texttt{nf}\,(\widehat{\sigma}) : UC$.

B.4.11 Positive Subtyping

Observation 12 (Positive Subtyping is Deterministic). *For fixed* Θ, Ξ, P, *and* Q, *if* $\Theta; \Xi \vDash P \geqslant Q \dashv C$ *and* $\Theta; \Xi \vDash P \geqslant Q \dashv C'$ *then* $C = C'$.

Lemma 86 (Soundness of the Positive Subtyping). *If* $\Theta \vdash^{\supseteq} \Xi$, $\Theta \vdash Q$, $\Theta\,;\mathtt{dom}\,(\Xi) \vdash P$, *and* $\Theta;\ \Xi \vDash P \geqslant Q \dashv C$, *then* $\Xi \vdash C : \mathtt{fav} P$ *and for any normalized* $\widehat{\sigma}$ *such that* $\Xi \vdash \widehat{\sigma} : C$, $\Theta \vdash [\widehat{\sigma}]P \geqslant Q$.

Lemma 87 (Completeness of the Positive Subtyping). *Suppose that* $\Theta \vdash^{\supseteq} \Xi$, $\Theta \vdash Q$ *and* $\Theta\,;\mathtt{dom}\,(\Xi) \vdash P$. *Then for any* $\Xi \vdash \widehat{\sigma} : \mathtt{fav}(P)$ *such that* $\Theta \vdash [\widehat{\sigma}]P \geqslant Q$, *there exists* $\Theta;\ \Xi \vDash P \geqslant Q \dashv C$ *and moreover,* $\Xi \vdash \widehat{\sigma} : C$.

B.4.12 Subtyping Constraint Merge

Observation 13 (Positive Subtyping is Deterministic). *For fixed* Θ, Ξ, P, *and* Q, *if* $\Theta;\ \Xi \vDash P \geqslant Q \dashv C$ *and* $\Theta;\ \Xi \vDash P \geqslant Q \dashv C'$ *then* $C = C'$.

Observation 14 (Constraint Entry Merge is Deterministic). *For fixed* Θ, e_1, e_2, *if* $\Theta \vdash e_1 \,\&\, e_2 = e$ *and* $\Theta \vdash e_1 \,\&\, e_2 = e'$ *then* $e = e'$.

Observation 15 (Subtyping Constraint Merge is Deterministic). *Suppose that* $\Xi \vdash C_1$ *and* $\Xi \vdash C_2$ *If* $\Xi \vdash C_1 \,\&\, C_2 = C$ *and* $\Xi \vdash C_1 \,\&\, C_2 = C'$ *are defined then* $C = C'$.

Lemma 88 (Soundness of Constraint Entry Merge). *For a fixed context* Θ, *suppose that* $\Theta \vdash e_1$ *and* $\Theta \vdash e_2$. *If* $\Theta \vdash e_1 \,\&\, e_2 = e$ *is defined then*

1. $\Theta \vdash e$
2. *For any* $\Theta \vdash P$, $\Theta \vdash P : e$ *implies* $\Theta \vdash P : e_1$ *and* $\Theta \vdash P : e_2$

Lemma 89 (Soundness of Constraint Merge). *Suppose that* $\Xi \vdash C_1 : \widehat{\Theta}_1$ *and* $\Xi \vdash C_2 : \widehat{\Theta}_2$ *and* $\Xi \vdash C_1 \,\&\, C_2 = C$ *is defined. Then*

1. $\Xi \vdash C : \widehat{\Theta}_1 \cup \widehat{\Theta}_2$,
2. *for any substitution* $\Xi \vdash \widehat{\sigma} : \widehat{\Theta}_1 \cup \widehat{\Theta}_2$, $\Xi \vdash \widehat{\sigma} : C$ *implies* $\Xi \vdash \widehat{\sigma} : C_1$ *and* $\Xi \vdash \widehat{\sigma} : C_2$.

Lemma 90 (Completeness of Constraint Entry Merge). *For a fixed context* Θ, *suppose that* $\Theta \vdash e_1$ *and* $\Theta \vdash e_2$ *are matching constraint entries.*

- *for a type* P *such that* $\Theta \vdash P : e_1$ *and* $\Theta \vdash P : e_2$, $\Theta \vdash e_1 \,\&\, e_2 = e$ *is defined and* $\Theta \vdash P : e$.
- *for a type* N *such that* $\Theta \vdash N : e_1$ *and* $\Theta \vdash N : e_2$, $\Theta \vdash e_1 \,\&\, e_2 = e$ *is defined and* $\Theta \vdash N : e$.

Lemma 91 (Completeness of Constraint Merge). *Suppose that* $\Xi \vdash C_1 : \widehat{\Theta}_1$ *and* $\Xi \vdash C_2 : \widehat{\Theta}_2$. *If there exists a substitution* $\Xi \vdash \widehat{\sigma} : \widehat{\Theta}_1 \cup \widehat{\Theta}_2$ *such that* $\Xi \vdash \widehat{\sigma} : C_1$ *and* $\Xi \vdash \widehat{\sigma} : C_2$ *then* $\Xi \vdash C_1 \,\&\, C_2 = C$ *is defined.*

B.4.13 Negative Subtyping

Observation 16 (Negative Algorithmic Subtyping is Deterministic). *For fixed* Θ, Ξ, M, *and* N, *if* $\Theta;\ \Xi \vDash N \leqslant M \dashv C$ *and* $\Theta;\ \Xi \vDash N \leqslant M \dashv C'$ *then* $C = C'$.

Lemma 92 (Soundness of Negative Subtyping). *If* $\Theta \vdash^{\supseteq} \Xi$, $\Theta \vdash M$, $\Theta\,;\mathtt{dom}\,(\Xi) \vdash N$ *and* $\Theta;\ \Xi \vDash N \leqslant M \dashv C$, *then* $\Xi \vdash C : \mathtt{fav}(N)$ *and for any normalized* $\widehat{\sigma}$ *such that* $\Xi \vdash \widehat{\sigma} : C$, $\Theta \vdash [\widehat{\sigma}]N \leqslant M$.

Lemma 93 (Completeness of the Negative Subtyping). *Suppose that* $\Theta \vdash^{\supseteq} \Xi$, $\Theta \vdash M$, $\Theta\,;\mathtt{dom}\,(\Xi) \vdash N$, *and* N *does not contain negative unification variables* ($\widehat{\alpha}^- \notin \mathtt{fav} N$). *Then for any* $\Xi \vdash \widehat{\sigma} : \mathtt{fav}(N)$ *such that* $\Theta \vdash [\widehat{\sigma}]N \leqslant M$, *there exists* $\Theta;\ \Xi \vDash N \leqslant M \dashv C$ *and moreover,* $\Xi \vdash \widehat{\sigma} : C$.

B.4.14 Singularity and Minimal Instantiation

Lemma 94 (Soundness of Minimal Instantiation)**.** *Suppose that* $\Theta \vdash^{\supseteq} \Xi$, $\Xi \vdash C$, *and* $\Theta\,;\mathsf{dom}\,(\Xi) \vdash P$. *If* P *is* C*-minimized by* $\widehat{\sigma}$ *then*

- $\Xi \vdash \widehat{\sigma} : \mathtt{fav}\,P$,
- $\Xi \vdash \widehat{\sigma} : C$,
- $\widehat{\sigma}$ *is normalized, and*
- *for any other* $\Xi \vdash \widehat{\sigma}' : \mathtt{fav}\,P$ *respecting* C *(i.e.,* $\Xi \vdash \widehat{\sigma}' : C$*), we have* $\Theta \vdash [\widehat{\sigma}']P \geqslant [\widehat{\sigma}]P$.

Lemma 95 (Completeness of Minimal Instantiation)**.** *Suppose that* $\Theta \vdash^{\supseteq} \Xi$, $\Xi \vdash C$, $\Theta\,;\mathsf{dom}\,(\Xi) \vdash P$, *and there exists* $\Xi \vdash \widehat{\sigma} : \mathtt{fav}\,P$ *respecting* C $(\Xi \vdash \widehat{\sigma} : C)$ *such that for any other* $\Xi \vdash \widehat{\sigma}' : \mathtt{fav}\,P$ *respecting* C $(\Xi \vdash \widehat{\sigma}' : C)$, *we have* $\Theta \vdash [\widehat{\sigma}']P \geqslant [\widehat{\sigma}]P$. *Then* P *is* C*-minimized by* $\mathtt{nf}\,(\widehat{\sigma})$.

Observation 17 (Minimal Instantiation is Deterministic)**.** *Suppose that* $\Theta \vdash^{\supseteq} \Xi$, $\Xi \vdash C$, $\Theta\,;\mathsf{dom}\,(\Xi) \vdash P$. *Then* P *is* C*-minimized by* $\widehat{\sigma}$ *and* P *is* C*-minimized by* $\widehat{\sigma}'$ *implies* $\widehat{\sigma} = \widehat{\sigma}'$.

Lemma 96 (Soundness of Entry Singularity)**.**

+ *Suppose* $e\ \mathtt{singular\ with}\ P$ *for* P *well-formed in* Θ. *Then* $\Theta \vdash P : e$, P *is normalized, and for any* $\Theta \vdash P'$ *such that* $\Theta \vdash P' : e$, $\Theta \vdash P' \simeq^{\leqslant} P$;
- *Suppose* $e\ \mathtt{singular\ with}\ N$ *for* N *well-formed in* Θ. *Then* $\Theta \vdash N : e$, N *is normalized, and for any* $\Theta \vdash N'$ *such that* $\Theta \vdash N' : e$, $\Theta \vdash N' \simeq^{\leqslant} N$.

Lemma 97 (Completeness of Entry Singularity)**.**

- *Suppose that there exists* N *well-formed in* Θ *such that for any* N' *well-formed in* Θ, $\Theta \vdash N' : e$ *implies* $\Theta \vdash N' \simeq^{\leqslant} N$. *Then* $e\ \mathtt{singular\ with\ nf}\ (N)$.
+ *Suppose that there exists* P *well-formed in* Θ *such that for any* P' *well-formed in* Θ, $\Theta \vdash P' : e$ *implies* $\Theta \vdash P' \simeq^{\leqslant} P$. *Then* $e\ \mathtt{singular\ with\ nf}\ (P)$.

Lemma 98 (Soundness of Singularity)**.** *Suppose* $\Xi \vdash C : \widehat{\Theta}$, *and* $C\ \mathtt{singular\ with}\ \widehat{\sigma}$. *Then* $\Xi \vdash \widehat{\sigma} : \widehat{\Theta}$, $\Xi \vdash \widehat{\sigma} : C$, $\widehat{\sigma}$ *is normalized, and for any* $\widehat{\sigma}'$ *such that* $\Xi \vdash \widehat{\sigma} : C$, $\Xi \vdash \widehat{\sigma}' \simeq^{\leqslant} \widehat{\sigma} : \widehat{\Theta}$.

Observation 18 (Singularity is Deterministic)**.** *For a fixed* C *such that* $\Xi \vdash C : \widehat{\Theta}$, *if* $C\ \mathtt{singular\ with}\ \widehat{\sigma}$ *and* $C\ \mathtt{singular\ with}\ \widehat{\sigma}'$, *then* $\widehat{\sigma} = \widehat{\sigma}'$.

Lemma 99 (Completeness of Singularity)**.** *For a given* $\Xi \vdash C$, *suppose that all the substitutions satisfying* C *are equivalent on* $\widehat{\Theta} \supseteq \mathsf{dom}\,(C)$. *In other words, suppose that there exists* $\Xi \vdash \widehat{\sigma}_1 : \widehat{\Theta}$ *such that for any* $\Xi \vdash \widehat{\sigma} : \widehat{\Theta}$, $\Xi \vdash \widehat{\sigma} : C$ *implies* $\Xi \vdash \widehat{\sigma} \simeq^{\leqslant} \widehat{\sigma}_1 : \widehat{\Theta}$. *Then*

- $C\ \mathtt{singular\ with}\ \widehat{\sigma}_0$ *for some* $\widehat{\sigma}_0$ *and*
- $\widehat{\Theta} = \mathsf{dom}\,(C)$.

B.4.15 Correctness of the Typing Algorithm

Lemma 100 (Determinacy of typing algorithm)**.** *Suppose that* $\Theta \vdash \Gamma$ *and* $\Theta \vdash^{\supseteq} \Xi$. *Then*

+ *If* $\Theta;\Gamma \vDash v\colon P$ *and* $\Theta;\Gamma \vDash v\colon P'$ *then* $P = P'$.
- *If* $\Theta;\Gamma \vDash c\colon N$ *and* $\Theta;\Gamma \vDash c\colon N'$ *then* $N = N'$.

- *If* $\Theta\,;\Gamma\,;\Xi \vDash N \bullet \vec{V} \Longrightarrow M \dashv \Xi'\,; C$ *and* $\Theta\,;\Gamma\,;\Xi \vDash N \bullet \vec{V} \Longrightarrow M' \dashv \Xi'\,; C'$ *then* $M = M'$, $\Xi = \Xi'$, *and* $C = C'$.

Lemma 101 (Soundness of typing)**.** *Suppose that* $\Theta \vdash \Gamma$. *For an inference tree* T_1,

+ *If* T_1 *infers* $\Theta; \Gamma \vDash v\colon P$ *then* $\Theta \vdash P$ *and* $\Theta; \Gamma \vdash v\colon P$
- *If* T_1 *infers* $\Theta; \Gamma \vDash c\colon N$ *then* $\Theta \vdash N$ *and* $\Theta; \Gamma \vdash c\colon N$
- *If* T_1 *infers* $\Theta\,;\Gamma\,;\Xi \vDash N \bullet \vec{V} \Longrightarrow M \dashv \Xi'\,; C$ *for* $\Theta \vdash^{\supseteq} \Xi$ *and* $\Theta\,;\mathtt{dom}\,(\Xi) \vdash N$ *free from negative algorithmic variables, then*
  1. $\Theta \vdash^{\supseteq} \Xi'$
  2. $\Xi \subseteq \Xi'$
  3. $\Theta\,;\mathtt{dom}\,(\Xi') \vdash M$
  4. $\mathtt{dom}\,(\Xi) \cap \mathtt{fav}(M) \subseteq \mathtt{fav} N$
  5. M *is normalized and free from negative algorithmic variables*
  6. $\Xi'|_{\mathtt{fav} N \cup \mathtt{fav} M} \vdash C$
  7. *for any* $\Xi' \vdash \widehat{\sigma} : \mathtt{fav} N \cup \mathtt{fav} M, \Xi' \vdash \widehat{\sigma} : C$ *implies* $\Theta\,;\Gamma \vdash [\widehat{\sigma}]N \bullet \vec{V} \Longrightarrow [\widehat{\sigma}]M$

Lemma 102 (Completeness of Typing)**.** *Suppose that* $\Theta \vdash \Gamma$. *For an inference tree* T_1,

+ *If* T_1 *infers* $\Theta; \Gamma \vdash v\colon P$ *then* $\Theta; \Gamma \vDash v\colon \mathtt{nf}\,(P)$
- *If* T_1 *infers* $\Theta; \Gamma \vdash c\colon N$ *then* $\Theta; \Gamma \vDash c\colon \mathtt{nf}\,(N)$
- *If* T_1 *infers* $\Theta\,;\Gamma \vdash [\widehat{\sigma}]N \bullet \vec{V} \Longrightarrow M$ *and*
  1. $\Theta \vdash^{\supseteq} \Xi$,
  2. $\Theta \vdash M$,
  3. $\Theta\,;\mathtt{dom}\,(\Xi) \vdash N$ *(free from negative algorithmic variables, that is* $\widehat{\alpha}^- \notin \mathtt{fav} N$*), and*
  4. $\Xi \vdash \widehat{\sigma} : \mathtt{fav}(N)$,

  then there exist M', Ξ', *and* C *such that*
  1. $\Theta\,;\Gamma\,;\Xi \vDash N \bullet \vec{V} \Longrightarrow M' \dashv \Xi'\,; C$ *and*
  2. *for any* $\Xi \vdash \widehat{\sigma} : \mathtt{fav}(N)$ *and* $\Theta \vdash M$ *such that* $\Theta\,;\Gamma \vdash [\widehat{\sigma}]N \bullet \vec{V} \Longrightarrow M$, *there exists* $\widehat{\sigma}'$ *such that*
     a. $\Xi' \vdash \widehat{\sigma}' : \mathtt{fav} N \cup \mathtt{fav} M'$ *and* $\Xi' \vdash \widehat{\sigma}' : C$,
     b. $\Xi \vdash \widehat{\sigma}' \simeq^{\leqslant} \widehat{\sigma} : \mathtt{fav} N$, *and*
     c. $\Theta \vdash [\widehat{\sigma}']M' \simeq^{\leqslant} M$.

Appendix C Theorem Proofs

C.1 Declarative Types

C.1.1 Type Well-Formedness

Lemma 3 (Soundness of type well-formedness)**.**

+ *If* $\Theta \vdash P$ *then* $\mathtt{fv}\,(P) \subseteq \Theta$,
− *if* $\Theta \vdash N$ *then* $\mathtt{fv}\,(N) \subseteq \Theta$.

Proof The proof is done by a simple structural induction on $\Theta \vdash P$ and mutually, $\Theta \vdash N$.

Case 1. $\Theta \vdash \alpha^{\pm}$ means by inversion that $\alpha^{\pm} \in \Theta$, that is, $\alpha^{\pm} = \mathtt{fv}\,(\alpha^{\pm}) \subseteq \Theta$.

Case 2. $\Theta \vdash Q \to M$ means by inversion that $\Theta \vdash Q$ and $\Theta \vdash M$. Then by the induction hypothesis, $\mathtt{fv}\,(Q) \subseteq \Theta$ and $\mathtt{fv}\,(M) \subseteq \Theta$, and hence, $\mathtt{fv}\,(Q \to M) = \mathtt{fv}\,(Q) \cup \mathtt{fv}\,(M) \subseteq \Theta$.

Case 3. the cases when $P = {\downarrow}N'$ or $N = {\uparrow}P'$ are proven analogously.

Case 4. $\Theta \vdash \forall\overrightarrow{\alpha^{+}}.\, M$ means by inversion that $\Theta, \overrightarrow{\alpha^{+}} \vdash M$. Then by the induction hypothesis, $\mathtt{fv}\,(M) \subseteq \Theta, \overrightarrow{\alpha^{+}}$, and hence, $\mathtt{fv}\,(\forall\overrightarrow{\alpha^{+}}.\, M) = \mathtt{fv}\,(M) \setminus \overrightarrow{\alpha^{+}} \subseteq \Theta, \overrightarrow{\alpha^{+}} \setminus \overrightarrow{\alpha^{+}} = \Theta$.

Case 5. The case $P = \exists\overrightarrow{\alpha^{-}}.\, Q$ is proven analogously.

■

Lemma 4 (Completeness of type well-formedness)**.** *In the well-formedness judgment, only used variables matter:*

+ *if* $\Theta_1 \cap \mathtt{fv}\,P = \Theta_2 \cap \mathtt{fv}\,P$ *then* $\Theta_1 \vdash P \iff \Theta_2 \vdash P$,
− *if* $\Theta_1 \cap \mathtt{fv}\,N = \Theta_2 \cap \mathtt{fv}\,N$ *then* $\Theta_1 \vdash N \iff \Theta_2 \vdash N$.

Proof By simple mutual induction on P and N. ■

Corollary 1 (Context Strengthening)**.**

+ *If* $\Theta \vdash P$ *then* $\mathtt{fv}\,(P) \vdash P$;
− *If* $\Theta \vdash N$ *then* $\mathtt{fv}\,(N) \vdash N$.

Proof It follows from Lemma 4 and Lemma 3.

+ By Lemma 3, $\mathtt{fv}\,(P) \subseteq \Theta$, and hence, $\Theta \cap \mathtt{fv}\,P = \mathtt{fv}\,P$, which makes Lemma 4 applicable fore contexts Θ and $\mathtt{fv}\,(P)$.
− The negative case is proven analogously.

■

Corollary 2 (Well-formedness Context Weakening)**.** *Suppose that* $\Theta_1 \subseteq \Theta_2$, *then*

+ *if* $\Theta_1 \vdash P$ *then* $\Theta_2 \vdash P$,

– *if* $\Theta_1 \vdash N$ *then* $\Theta_2 \vdash N$.

Proof By Lemma 3, $\Theta_1 \vdash P$ implies $\mathtt{fv}\,(P) \subseteq \Theta_1$, which means that $\mathtt{fv}\,(P) \subseteq \Theta_2$, and thus, $\mathtt{fv}\,(P) = \mathtt{fv}\,(P) \cap \Theta_1 = \mathtt{fv}\,(P) \cap \Theta_2$. Then by Lemma 4, $\Theta_2 \vdash P$. The negative case is symmetric. ∎

Lemma 5 (Well-formedness agrees with substitution)**.** *Suppose that* $\Theta_2 \vdash \sigma : \Theta_1$. *Then*

+ $\Theta, \Theta_1 \vdash P$ *implies* $\Theta, \Theta_2 \vdash [\sigma]P$, *and*

– $\Theta, \Theta_1 \vdash N$ *implies* $\Theta, \Theta_2 \vdash [\sigma]N$.

Proof We prove it by induction on $\Theta, \Theta_1 \vdash P$ and mutually, on $\Theta, \Theta_1 \vdash N$. Let us consider the last rule used in the derivation.

Case 1. $(\textsc{Var}_+^{\mathrm{WF}})$, i.e. P is α^+.
By inversion, $\alpha^+ \in \Theta, \Theta_1$, then

- if $\alpha^+ \in \Theta_1$ then $\Theta_2 \vdash [\sigma]\alpha^+$, and by weakening (Corollary 2), $\Theta, \Theta_2 \vdash [\sigma]\alpha^+$;
- if $\alpha^+ \in \Theta \setminus \Theta_1$ then $[\sigma]\alpha^+ = \alpha^+$, and by $(\textsc{Var}_+^{\mathrm{WF}})$, $\Theta, \Theta_2 \vdash \alpha^+$.

Case 2. $(\uparrow^{\mathrm{WF}})$, i.e. P is $\downarrow N$.
Then $\Theta, \Theta_1 \vdash \downarrow N$ means $\Theta, \Theta_1 \vdash N$ by inversion, and by the induction hypothesis, $\Theta, \Theta_2 \vdash [\sigma]N$. Then by $(\uparrow^{\mathrm{WF}})$, $\Theta, \Theta_2 \vdash \downarrow[\sigma]N$, which by definition of substitution is rewritten as $\Theta, \Theta_2 \vdash [\sigma]\downarrow N$.

Case 3. $(\exists^{\mathrm{WF}})$, i.e. P is $\exists\overrightarrow{\alpha^-}.\, Q$.
Then $\Theta, \Theta_1 \vdash \exists\overrightarrow{\alpha^-}.\, Q$ means $\Theta, \overrightarrow{\alpha^-}, \Theta_1 \vdash Q$ by inversion, and by the induction hypothesis, $\Theta, \overrightarrow{\alpha^-}, \Theta_2 \vdash [\sigma]Q$. Then by $(\exists^{\mathrm{WF}})$, $\Theta, \overrightarrow{\alpha^-}, \Theta_2 \vdash \exists\overrightarrow{\alpha^-}.\, [\sigma]Q$, which by definition of substitution is rewritten as $\Theta, \Theta_2 \vdash [\sigma]\exists\overrightarrow{\alpha^-}.\, Q$.

Case 4. The negative cases are proved symmetrically.

∎

C.1.2 Substitution

Lemma 6 (Substitution strengthening)**.** *Restricting the substitution to the free variables of the substitution subject does not affect the result. Suppose that* σ *is a substitution,* P *and* N *are types. Then*

+ $[\sigma]P = [\sigma|_{\mathtt{fv}\,P}]P$,

– $[\sigma]N = [\sigma|_{\mathtt{fv}\,N}]N$

Proof First, we strengthen the statement by saying that one can restrict the substitution to an arbitrary superset of the free variables of the substitution subject:

+ $[\sigma]P = [\sigma|_{vars}]P$, for any $vars \supseteq \mathtt{fv}\,P$, and

– $[\sigma]N = [\sigma|_{vars}]N$, for any $vars \supseteq \mathtt{fv}\,N$.

Then the proof is a straightforward induction on P and mutually, on N. For the base cases:

Case 1. $N = \alpha^-$

Then $[\sigma]\alpha^- = \sigma|_{vars}(\alpha^-)$ by definition, since $\alpha^- \in \mathtt{fv}\ \alpha^- \subseteq vars$.

Case 2. $N = P \to M$

Then $[\sigma](P \to M) = [\sigma]P \to [\sigma]M$ by definition. Since $\mathtt{fv}\ P \subseteq \mathtt{fv}\ (P \to M) \subseteq vars$, the induction hypothesis is applicable to $[\sigma]P$: $[\sigma]P = [\sigma|_{vars}]P$. Analogously, and $[\sigma]M = [\sigma|_{vars}]M$. Then $[\sigma](P \to M) = [\sigma|_{vars}]P \to [\sigma|_{vars}]M = [\sigma|_{vars}](P \to M)$.

Case 3. $N = \uparrow P$ is proved analogously to the previous case.

Case 4. $N = \forall\overrightarrow{\alpha^\pm}.\ M$ (where $\overrightarrow{\alpha^\pm}$ is not empty)

Then $[\sigma]\forall\overrightarrow{\alpha^\pm}.\ M = \forall\overrightarrow{\alpha^\pm}.\ [\sigma]M$ by definition. Let us assume $\overrightarrow{\alpha^\pm}$ are fresh variables, it means that $\sigma(\alpha^\pm) = \alpha^\pm$ for any $\alpha^\pm \in \overrightarrow{\alpha^\pm}$, and thus, $\sigma|_{vars} = \sigma|_{(vars \cup \overrightarrow{\alpha^\pm})}$ immediately from the definition.

Since $vars \subseteq \mathtt{fv}\ (\forall\overrightarrow{\alpha^\pm}.\ M) = \mathtt{fv}\ M \setminus \overrightarrow{\alpha^\pm}$, $vars \cup \overrightarrow{\alpha^\pm} \subseteq \mathtt{fv}\ (M)$. Then by the induction hypothesis, $[\sigma]M = [\sigma|_{(vars \cup \overrightarrow{\alpha^\pm})}]M$. Finally, $[\sigma]\forall\overrightarrow{\alpha^\pm}.\ M = \forall\overrightarrow{\alpha^\pm}.\ [\sigma|_{(vars \cup \overrightarrow{\alpha^\pm})}]M = \forall\overrightarrow{\alpha^\pm}.\ [\sigma|_{vars}]M = [\sigma|_{vars}]\forall\overrightarrow{\alpha^\pm}.\ M$.

Case 5. The positive cases are proven symmetrically.

■

Lemma 7 (Signature of a restricted substitution)**.** *If* $\Theta_2 \vdash \sigma : \Theta_1$ *then* $\Theta_2 \vdash \sigma|_{vars} : \Theta_1 \cap vars$.

Proof Let us take an arbitrary $\alpha^\pm \in \Theta_1 \cap vars$. Since $\alpha^\pm \in \Theta_1$, $\Theta_2 \vdash [\sigma]\alpha^\pm$ by the signature of σ.

Let us take an arbitrary $\alpha^\pm \notin \Theta_1 \cap vars$. If $\alpha^\pm \notin vars$ then $[\sigma|_{vars}]\alpha^\pm = \alpha^\pm$ by definition of restriction. If $\alpha^\pm \in vars \setminus \Theta_1$ then $[\sigma|_{vars}]\alpha^\pm = [\sigma]\alpha^\pm$ by definition and $[\sigma]\alpha^\pm = \alpha^\pm$ by the signature of σ. ■

Lemma 8. *Suppose that* σ *is a substitution with signature* $\Theta_2 \vdash \sigma : \Theta_1$. *Then if vars is disjoint from* Θ_1, *then* $\sigma|_{vars} = \mathtt{id}$.

Proof Let us take an arbitrary $\alpha^\pm$. If $\alpha^\pm \notin vars$ then $[\sigma|_{vars}]\alpha^\pm = \alpha^\pm$ by definition.

If $\alpha^\pm \in vars$ then $\alpha^\pm \notin \Theta_1$ by assumption. Then $[\sigma|_{vars}]\alpha^\pm = [\sigma]\alpha^\pm$ by definition of restricted substitution, and since $\Theta_2 \vdash \sigma : \Theta_1$, we have $[\sigma]\alpha^\pm = \alpha^\pm$. ■

Corollary 3 (Application of a disjoint substitution)**.** *Suppose that* σ *is a substitution with signature* $\Theta_2 \vdash \sigma : \Theta_1$. *Then*

+ *if* $\Theta_1 \cap \mathtt{fv}\ (Q) = \emptyset$ *then* $[\sigma]Q = Q$;
− *if* $\Theta_1 \cap \mathtt{fv}\ (N) = \emptyset$ *then* $[\sigma]N = N$.

Lemma 9 (Substitution range weakening)**.** *Suppose that* $\Theta_2 \subseteq \Theta_2'$ *are contexts and* σ *is a substitution. Then* $\Theta_2 \vdash \sigma : \Theta_1$ *implies* $\Theta_2' \vdash \sigma : \Theta_1$.

Proof For any $\alpha^\pm \in \Theta_1$, $\Theta_2 \vdash \sigma : \Theta_1$ gives us $\Theta_2 \vdash [\sigma]\alpha^\pm$, which can be weakened to $\Theta_2' \vdash [\sigma]\alpha^\pm$ by Corollary 2. This way, $\Theta_2' \vdash \sigma : \Theta_1$. ■

Lemma 10 (Subsitutions Equivalent on Free Variables)**.** *Suppose that* $\Theta' \subseteq \Theta$*,* σ_1 *and* σ_2 *are substitutions of signature* $\Theta \vdash \sigma_i : \Theta'$*. Then*

+ *for a type* $\Theta \vdash P$*, if* $\Theta \vdash [\sigma_1]P \simeq^{\leqslant} [\sigma_2]P$ *then* $\Theta \vdash \sigma_1 \simeq^{\leqslant} \sigma_2 : \mathtt{fv}\, P \cap \Theta'$*;*

− *for a type* $\Theta \vdash N$*, if* $\Theta \vdash [\sigma_1]N \simeq^{\leqslant} [\sigma_2]N$ *then* $\Theta \vdash \sigma_1 \simeq^{\leqslant} \sigma_2 : \mathtt{fv}\, N \cap \Theta'$*.*

Proof Let us make an additional assumption that σ_1, σ_2, and the mentioned types are normalized. If they are not, we normalize them first.

Notice that the normalization preserves the set of free variables (Lemma 40), well-formedness (Corollary 14), and equivalence (Lemma 48), and distributes over substitution (Lemma 43). This way, the assumed and desired properties are equivalent to their normalized versions.

We prove it by induction on the structure of P and mutually, N. Let us consider the shape of this type.

Case 1. $P = \alpha^+ \in \Theta'$. Then $\Theta \vdash \sigma_1 \simeq^{\leqslant} \sigma_2 : \mathtt{fv}\, P \cap \Theta'$ means $\Theta \vdash \sigma_1 \simeq^{\leqslant} \sigma_2 : \alpha^+$, i.e. $\Theta \vdash [\sigma_1]\alpha^+ \simeq^{\leqslant} [\sigma_2]\alpha^+$, which holds by assumption.

Case 2. $P = \alpha^+ \in \Theta \setminus \Theta'$. Then $\mathtt{fv}\, P \cap \Theta' = \cdot$, so $\Theta \vdash \sigma_1 \simeq^{\leqslant} \sigma_2 : \mathtt{fv}\, P \cap \Theta'$ holds vacuously.

Case 3. $P = {\downarrow}N$. Then the induction hypothesis is applicable to type N:

1. N is normalized,
2. $\Theta \vdash N$ by inversion of $\Theta \vdash {\downarrow}N$,
3. $\Theta \vdash [\sigma_1]N \simeq^{\leqslant} [\sigma_2]N$ holds by inversion of $\Theta \vdash [\sigma_1]{\downarrow}N \simeq^{\leqslant} [\sigma_2]{\downarrow}N$, i.e. $\Theta \vdash {\downarrow}[\sigma_1]N \simeq^{\leqslant} {\downarrow}[\sigma_2]N$.

This way, we obtain $\Theta \vdash \sigma_1 \simeq^{\leqslant} \sigma_2 : \mathtt{fv}\, N \cap \Theta'$, which implies the required equivalence since $\mathtt{fv}\, P \cap \Theta' = \mathtt{fv}\, {\downarrow}N \cap \Theta' = \mathtt{fv}\, N \cap \Theta'$.

Case 4. $P = \exists\overrightarrow{\alpha^-}.\, Q$ Then the induction hypothesis is applicable to type Q well-formed in context $\Theta, \overrightarrow{\alpha^-}$:

1. $\Theta' \subseteq \Theta, \overrightarrow{\alpha^-}$ since $\Theta' \subseteq \Theta$,
2. $\Theta, \overrightarrow{\alpha^-} \vdash \sigma_i : \Theta'$ by weakening,
3. Q is normalized,
4. $\Theta, \overrightarrow{\alpha^-} \vdash Q$ by inversion of $\Theta \vdash \exists\overrightarrow{\alpha^-}.\, Q$,
5. Notice that $[\sigma_i]\exists\overrightarrow{\alpha^-}.\, Q$ is normalized, and thus, $[\sigma_1]\exists\overrightarrow{\alpha^-}.\, Q \simeq^{D} [\sigma_2]\exists\overrightarrow{\alpha^-}.\, Q$ implies $[\sigma_1]\exists\overrightarrow{\alpha^-}.\, Q = [\sigma_2]\exists\overrightarrow{\alpha^-}.\, Q$ (by Lemma 48).). This equality means $[\sigma_1]Q = [\sigma_2]Q$, which implies $\Theta \vdash [\sigma_1]Q \simeq^{\leqslant} [\sigma_2]Q$.

Case 5. $N = P \to M$

■

Lemma 11 (Substitution composition well-formedness). *If* $\Theta_1' \vdash \sigma_1 : \Theta_1$ *and* $\Theta_2' \vdash \sigma_2 : \Theta_2$, *then* $\Theta_1', \Theta_2' \vdash \sigma_2 \circ \sigma_1 : \Theta_1, \Theta_2$.

Proof By definition of composition. ∎

Lemma 12 (Substitution monadic composition well-formedness). *If* $\Theta_1' \vdash \sigma_1 : \Theta_1$ *and* $\Theta_2' \vdash \sigma_2 : \Theta_2$, *then* $\Theta_2' \vdash \sigma_2 \ll \sigma_1 : \Theta_1$.

Proof By definition of monadic composition. ∎

Lemma 13 (Substitution composition). *If* $\Theta_1' \vdash \sigma_1 : \Theta_1$, $\Theta_2' \vdash \sigma_2 : \Theta_2$, $\Theta_1 \cap \Theta_2' = \emptyset$ *and* $\Theta_1 \cap \Theta_2 = \emptyset$ *then* $\sigma_2 \circ \sigma_1 = (\sigma_2 \ll \sigma_1) \circ \sigma_2$.

Proof

1. Suppose that $\alpha^\pm \in \Theta_1$ then $\alpha^\pm \notin \Theta_2$, and thus, $[(\sigma_2 \ll \sigma_1) \circ \sigma_2]\alpha^\pm = [(\sigma_2 \ll \sigma_1)]\alpha^\pm = [\sigma_2][\sigma_1]\alpha^\pm = [(\sigma_2 \circ \sigma_1)]\alpha^\pm$.
2. Suppose that $\alpha^\pm \notin \Theta_1$ then $[(\sigma_2 \circ \sigma_1)]\alpha^\pm = [\sigma_2]\alpha^\pm$. Then
   a. if $\alpha^\pm \notin \Theta_2$ then $[\sigma_2]\alpha^\pm = \alpha^\pm$ and $[(\sigma_2 \ll \sigma_1) \circ \sigma_2]\alpha^\pm = [(\sigma_2 \ll \sigma_1)][\sigma_2]\alpha^\pm = [\sigma_2 \ll \sigma_1]\alpha^\pm = \alpha^\pm$
   b. if $\alpha^\pm \in \Theta_2$ then $\Theta_2' \vdash [\sigma_2]\alpha^\pm$, and hence, $[(\sigma_2 \ll \sigma_1) \circ \sigma_2]\alpha^\pm = [(\sigma_2 \ll \sigma_1)][\sigma_2]\alpha^\pm = [\sigma_2]\alpha^\pm$ by definition of monadic composition, since none of the free variables of $[\sigma_2]\alpha^\pm$ is in Θ_1.

∎

Corollary 4 (Substitution composition commutativity). *If* $\Theta_1' \vdash \sigma_1 : \Theta_1$, $\Theta_2' \vdash \sigma_2 : \Theta_2$, *and* $\Theta_1 \cap \Theta_2 = \emptyset$, $\Theta_1 \cap \Theta_2' = \emptyset$, *and* $\Theta_1' \cap \Theta_2 = \emptyset$ *then* $\sigma_2 \circ \sigma_1 = \sigma_1 \circ \sigma_2$.

Proof by Lemma 13, $\sigma_2 \circ \sigma_1 = (\sigma_2 \ll \sigma_1) \circ \sigma_2$. Since the codomain of σ_1 is Θ_1', and it is disjoint with the domain of σ_2, $\sigma_2 \ll \sigma_1 = \sigma_1$. ∎

Lemma 14 (Substitution domain weakening). *If* $\Theta_2 \vdash \sigma : \Theta_1$ *then* $\Theta_2, \Theta' \vdash \sigma : \Theta_1, \Theta'$

Proof If the variable $\alpha^\pm$ is in Θ_1 then $\Theta_2 \vdash [\sigma]\alpha^\pm$ by assumption, and then $\Theta_2, \Theta' \vdash [\sigma]\alpha^\pm$ by weakening. If the variable $\alpha^\pm$ is in $\Theta' \setminus \Theta_1$ then $[\sigma]\alpha^\pm = \alpha^\pm \in \Theta' \subseteq \Theta_2, \Theta'$, and thus, $\Theta_2, \Theta' \vdash \alpha^\pm$. ∎

Lemma 15 (Free variables after substitution). *Suppose that* $\Theta_2 \vdash \sigma : \Theta_1$, *then*

+ *for a type* P, *the free variables of* $[\sigma]P$ *are bounded in the following way:* $\mathtt{fv}\,(P) \setminus \Theta_1 \subseteq \mathtt{fv}\,([\sigma]P) \subseteq (\mathtt{fv}\,(P) \setminus \Theta_1) \cup \Theta_2$;
− *for a type* N, *the free variables of* $[\sigma]P$ *are bounded in the following way:* $\mathtt{fv}\,(N) \setminus \Theta_1 \subseteq \mathtt{fv}\,([\sigma]N) \subseteq (\mathtt{fv}\,(N) \setminus \Theta_1) \cup \Theta_2$.

Proof We prove it by structural induction on P and mutually, on N.

Case 1. $P = \alpha^+$

If $\alpha^+ \in \Theta_1$ then $\Theta_2 \vdash [\sigma]\alpha^+$, and by Lemma 3, $\mathtt{fv}\,([\sigma]\alpha^+) \subseteq \Theta_2$. $\mathtt{fv}\,(\alpha^+) \setminus \Theta_1 = \cdot$, so $\mathtt{fv}\,([\sigma]P) \setminus \Theta_1 \subseteq \mathtt{fv}\,([\sigma]\alpha^+)$ vacuously.

If $\alpha^+ \notin \Theta_1$ then $[\sigma]\alpha^+ = \alpha^+$, and $\mathtt{fv}\,([\sigma]\alpha^+) = \alpha^+ = \alpha^+ \setminus \Theta_1$.

Case 2. $P = \exists\overrightarrow{\alpha^{\pm}}.\, Q$

Then we need to show that $\mathtt{fv}\,([\sigma]P) = \mathtt{fv}\,([\sigma]Q) \setminus \overrightarrow{\alpha^{\pm}}$ is a subset of $(\mathtt{fv}\,(P) \setminus \Theta_1) \cup \Theta_2$ and a superset of $\mathtt{fv}\,(P) \setminus \Theta_1$. Notice that $\mathtt{fv}\,(P) = \mathtt{fv}\,(Q) \setminus \overrightarrow{\alpha^{\pm}}$ by definition. This way, we need to show that $\mathtt{fv}\,(Q) \setminus \overrightarrow{\alpha^{\pm}} \setminus \Theta_1 \subseteq \mathtt{fv}\,([\sigma]Q) \setminus \overrightarrow{\alpha^{\pm}} \subseteq (\mathtt{fv}\,(Q) \setminus \overrightarrow{\alpha^{\pm}} \setminus \Theta_1) \cup \Theta_2$,

By the induction hypothesis, $\mathtt{fv}\,([\sigma]Q) \subseteq (\mathtt{fv}\,(Q) \setminus \Theta_1) \cup \Theta_2$. So for the second inclusion, it suffices to show that $((\mathtt{fv}\,(Q) \setminus \Theta_1) \cup \Theta_2) \setminus \overrightarrow{\alpha^{\pm}} \subseteq (\mathtt{fv}\,(Q) \setminus \overrightarrow{\alpha^{\pm}} \setminus \Theta_1) \cup \Theta_2$, which holds by set theoretical reasoning.

Also by the induction hypothesis, $\mathtt{fv}\,(Q) \setminus \Theta_1 \subseteq \mathtt{fv}\,([\sigma]Q)$, and thus, by subtracting $\overrightarrow{\alpha^{\pm}}$ from both sides, $\mathtt{fv}\,(Q) \setminus \overrightarrow{\alpha^{\pm}} \setminus \Theta_1 \subseteq \mathtt{fv}\,([\sigma]Q) \setminus \overrightarrow{\alpha^{\pm}}$.

Case 3. The case $N = \forall\overrightarrow{\alpha^{\pm}}.\, M$ is proved analogously.

Case 4. $N = P \to M$

Then $\mathtt{fv}\,([\sigma]N) = \mathtt{fv}\,([\sigma]P) \cup \mathtt{fv}\,([\sigma]M)$. By the induction hypothesis,

1. $\mathtt{fv}\,(P) \setminus \Theta_1 \subseteq \mathtt{fv}\,([\sigma]P) \subseteq (\mathtt{fv}\,(P) \setminus \Theta_1) \cup \Theta_2$ and
2. $\mathtt{fv}\,(M) \setminus \Theta_1 \subseteq \mathtt{fv}\,([\sigma]M) \subseteq (\mathtt{fv}\,(M) \setminus \Theta_1) \cup \Theta_2$.

We unite these inclusions vertically and obtain $\mathtt{fv}\,(P) \setminus \Theta_1 \cup \mathtt{fv}\,(M) \setminus \Theta_1 \subseteq \mathtt{fv}\,([\sigma]N) \subseteq ((\mathtt{fv}\,(P) \setminus \Theta_1) \cup \Theta_2) \cup ((\mathtt{fv}\,(M) \setminus \Theta_1) \cup \Theta_2)$, which is equivalent to $(\mathtt{fv}\,(P) \cup \mathtt{fv}\,(M)) \setminus \Theta_1 \subseteq \mathtt{fv}\,([\sigma]N) \subseteq (\mathtt{fv}\,(P) \cup \mathtt{fv}\,(M)) \setminus \Theta_1 \cup \Theta_2$. Since $\mathtt{fv}\,(P) \cup \mathtt{fv}\,(M) = \mathtt{fv}\,(N)$, $\mathtt{fv}\,(N) \setminus \Theta_1 \subseteq \mathtt{fv}\,([\sigma]N) \subseteq (\mathtt{fv}\,(N) \setminus \Theta_1) \cup \Theta_2$.

Case 5. The cases when $P = \downarrow M$ and $N = \uparrow Q$ are proved analogously

■

Lemma 16 (Free variables of a variable image)**.** *Suppose that σ is an arbitrary substitution, Then*

- $+$ *if* $\alpha^{\pm} \in \mathtt{fv}\,(P)$ *then* $\mathtt{fv}\,([\sigma]\alpha^{\pm}) \subseteq \mathtt{fv}\,([\sigma]P)$,
- $-$ *if* $\alpha^{\pm} \in \mathtt{fv}\,(N)$ *then* $\mathtt{fv}\,([\sigma]\alpha^{\pm}) \subseteq \mathtt{fv}\,([\sigma]N)$.

Proof By mutual induction on P and N. The base cases (when P or N is a variable) are trivial, since then $\alpha^{\pm} \in \mathtt{fv}\,(P)$ means $\alpha^{\pm} = P$ (and symmetrically for N). The congruent cases (when the type is formed by $\downarrow$, $\uparrow$, or $\to$) hold since $\alpha^{\pm}$ occurs in type means that it occurs in one of its parts, to which we apply the induction hypothesis.

Let us suppose that the type is $\exists\overrightarrow{\alpha^{\pm}}.\, Q$. Then $\alpha^{\pm} \in \mathtt{fv}\,(\exists\overrightarrow{\alpha^{\pm}}.\, Q)$ means $\alpha^{\pm} \in \mathtt{fv}\,(Q)$ and $\alpha^{\pm} \notin \overrightarrow{\alpha^{\pm}}$. Then by the induction hypothesis, $\mathtt{fv}\,([\sigma]\alpha^{\pm}) \subseteq \mathtt{fv}\,([\sigma]Q)$, and it is left to notice that $\mathtt{fv}\,([\sigma]\alpha^{\pm}) \cap \overrightarrow{\alpha^{\pm}} = \emptyset$, which we can ensure by alpha-equivalence. ■

C.1.3 Declarative Subtyping

Lemma 17 (Free Variable Propagation)**.** *In the judgments of negative subtyping or positive supertyping, free variables propagate left to right. For a context* Θ*,*

- $-$ *if* $\Theta \vdash N \leqslant M$ *then* $\mathtt{fv}\,(N) \subseteq \mathtt{fv}\,(M)$
- $+$ *if* $\Theta \vdash P \geqslant Q$ *then* $\mathtt{fv}\,(P) \subseteq \mathtt{fv}\,(Q)$

Proof Mutual induction on $\Theta \vdash N \leqslant M$ and $\Theta \vdash P \geqslant Q$.

Case 1. $\Theta \vdash \alpha^- \leqslant \alpha^-$
It is self-evident that $\alpha^- \subseteq \alpha^-$.

Case 2. $\Theta \vdash \uparrow P \leqslant \uparrow Q$ From the inversion (and unfolding $\Theta \vdash P \simeq^{\leqslant} Q$), we have $\Theta \vdash P \geqslant Q$. Then by the induction hypothesis, $\mathtt{fv}\,(P) \subseteq \mathtt{fv}\,(Q)$. The desired inclusion holds, since $\mathtt{fv}\,(\uparrow P) = \mathtt{fv}\,(P)$ and $\mathtt{fv}\,(\uparrow Q) = \mathtt{fv}\,(Q)$.

Case 3. $\Theta \vdash P \to N \leqslant Q \to M$ The induction hypothesis applied to the premises gives: $\mathtt{fv}\,(P) \subseteq \mathtt{fv}\,(Q)$ and $\mathtt{fv}\,(N) \subseteq \mathtt{fv}\,(M)$. Then $\mathtt{fv}\,(P \to N) = \mathtt{fv}\,(P) \cup \mathtt{fv}\,(N) \subseteq \mathtt{fv}\,(Q) \cup \mathtt{fv}\,(M) = \mathtt{fv}\,(Q \to M)$.

Case 4. $\Theta \vdash \forall \overrightarrow{\alpha^+}.\, N \leqslant \forall \overrightarrow{\beta^+}.\, M$

$$\begin{aligned} \mathtt{fv}\,\forall \overrightarrow{\alpha^+}.\, N &\subseteq \mathtt{fv}\,([\overrightarrow{P}/\overrightarrow{\alpha^+}]N) \setminus \overrightarrow{\beta^+} && \overrightarrow{\beta^+} \text{ is excluded by the premise } \mathtt{fv}\,N \cap \overrightarrow{\beta^+} = \emptyset \\ &\subseteq \mathtt{fv}\,M \setminus \overrightarrow{\beta^+} && \text{by the i.h., } \mathtt{fv}\,([\overrightarrow{P}/\overrightarrow{\alpha^+}]N) \subseteq \mathtt{fv}\,M \\ &\subseteq \mathtt{fv}\,\forall \overrightarrow{\beta^+}.\, M \end{aligned}$$

Case 5. The positive cases are symmetric.

■

Corollary 5 (Free Variables of mutual subtypes)**.**

- $-$ *If* $\Theta \vdash N \simeq^{\leqslant} M$ *then* $\mathtt{fv}\,N = \mathtt{fv}\,M$*,*
- $+$ *If* $\Theta \vdash P \simeq^{\leqslant} Q$ *then* $\mathtt{fv}\,P = \mathtt{fv}\,Q$

Corollary 6. *Suppose that all the types below are well-formed in* Θ *and* $\Theta' \subseteq \Theta$*. Then*

- $+$ $\Theta \vdash P \simeq^{\leqslant} Q$ *implies* $\Theta' \vdash P \iff \Theta' \vdash Q$
- $-$ $\Theta \vdash N \simeq^{\leqslant} M$ *implies* $\Theta' \vdash N \iff \Theta' \vdash M$

Proof From Lemma 4 and Corollary 5. ■

Lemma 18 (Decomposition of quantifier rules)**.** *Assuming that* $\overrightarrow{\alpha^+}$*,* $\overrightarrow{\beta^+}$*,* $\overrightarrow{\alpha^-}$*, and* $\overrightarrow{\alpha^-}$ *are disjoint from* Θ*,*

- $-_R$ $\Theta \vdash N \leqslant \forall \overrightarrow{\beta^+}.\, M$ *holds if and only if* $\Theta, \overrightarrow{\beta^+} \vdash N \leqslant M$*;*
- $+_R$ $\Theta \vdash P \geqslant \exists \overrightarrow{\beta^-}.\, Q$ *holds if and only if* $\Theta, \overrightarrow{\beta^-} \vdash P \geqslant Q$*;*
- $-_L$ *suppose* $M \neq \forall \ldots$ *then* $\Theta \vdash \forall \overrightarrow{\alpha^+}.\, N \leqslant M$ *holds if and only if* $\Theta \vdash [\overrightarrow{P}/\overrightarrow{\alpha^+}]N \leqslant M$ *for some* $\Theta \vdash \overrightarrow{P}$*;*

$+_L$ *suppose* $Q \neq \exists \ldots$ *then* $\Theta \vdash \exists \overrightarrow{\alpha^-}. P \geqslant Q$ *holds if and only if* $\Theta \vdash [\overrightarrow{N}/\overrightarrow{\alpha^-}]P \geqslant Q$ *for some* $\Theta \vdash \overrightarrow{N}$.

Proof

$-_R$ Let us prove both directions.

- $\Rightarrow$ Let us assume $\Theta \vdash N \leqslant \forall \overrightarrow{\beta^+}. M$. $\Theta \vdash N \leqslant \forall \overrightarrow{\beta^+}. M$. Let us decompose M as $\forall \overrightarrow{\beta^{+\prime}}. M'$ where M' does not start with $\forall$, and decompose N as $\forall \overrightarrow{\alpha^+}. N'$ where N' does not start with $\forall$. If $\overrightarrow{\beta^+}$ is empty, then $\Theta, \overrightarrow{\beta^+} \vdash N \leqslant M$ holds by assumption. Otherwise, $\Theta \vdash \forall \overrightarrow{\alpha^+}. N' \leqslant \forall \overrightarrow{\beta^+}. \forall \overrightarrow{\beta^{+\prime}}. M$ is inferred by $(\forall^{\leqslant})$, and by inversion: $\Theta, \overrightarrow{\beta^+}, \overrightarrow{\beta^{+\prime}} \vdash [\overrightarrow{P}/\overrightarrow{\alpha^+}]N' \leqslant M'$ for some $\Theta, \overrightarrow{\beta^+}, \overrightarrow{\beta^{+\prime}} \vdash \overrightarrow{P}$. Then again by $(\forall^{\leqslant})$ with the same $\overrightarrow{P}$, $\Theta, \overrightarrow{\beta^+} \vdash \forall \overrightarrow{\alpha^+}. N' \leqslant \forall \overrightarrow{\beta^{+\prime}}. M'$, that is $\Theta, \overrightarrow{\beta^+} \vdash N \leqslant M$.
- $\Leftarrow$ let us assume $\Theta, \overrightarrow{\beta^+} \vdash N \leqslant M$, and let us decompose N as $\forall \overrightarrow{\alpha^+}. N'$ where N' does not start with $\forall$, and M as $\forall \overrightarrow{\beta^{+\prime}}. M'$ where M' does not start with $\forall$. if $\overrightarrow{\alpha^+}$ and $\overrightarrow{\beta^{+\prime}}$ are empty then $\Theta, \overrightarrow{\beta^+} \vdash N \leqslant M$ is turned into $\Theta \vdash N \leqslant \forall \overrightarrow{\beta^+}. M$ by $(\forall^{\leqslant})$. Otherwise, $\Theta, \overrightarrow{\beta^+} \vdash \forall \overrightarrow{\alpha^+}. N' \leqslant \forall \overrightarrow{\beta^{+\prime}}. M'$ is inferred by $(\forall^{\leqslant})$, that is $\Theta, \overrightarrow{\beta^+}, \overrightarrow{\beta^{+\prime}} \vdash [\overrightarrow{P}/\overrightarrow{\alpha^+}]N' \leqslant M'$ for some $\Theta, \overrightarrow{\beta^+}, \overrightarrow{\beta^{+\prime}} \vdash \overrightarrow{P}$. Then by $(\forall^{\leqslant})$ again, $\Theta \vdash \forall \overrightarrow{\alpha^+}. N' \leqslant \forall \overrightarrow{\beta^+}, \overrightarrow{\beta^{+\prime}}. M'$, in other words, $\Theta \vdash \forall \overrightarrow{\alpha^+}. N' \leqslant \forall \overrightarrow{\beta^+}. \forall \overrightarrow{\beta^{+\prime}}. M'$, that is $\Theta \vdash N \leqslant \forall \overrightarrow{\beta^+}. M$.

$-_L$ Suppose $M \neq \forall \ldots$. Let us prove both directions.

- $\Rightarrow$ Let us assume $\Theta \vdash \forall \overrightarrow{\alpha^+}. N \leqslant M$. then if $\overrightarrow{\alpha^+} = \cdot$, $\Theta \vdash N \leqslant M$ holds immediately. Otherwise, let us decompose N as $\forall \overrightarrow{\alpha^{+\prime}}. N'$ where N' does not start with $\forall$. Then $\Theta \vdash \forall \overrightarrow{\alpha^+}. \forall \overrightarrow{\alpha^{+\prime}}. N' \leqslant M'$ is inferred by $(\forall^{\leqslant})$, and by inversion, there exist $\Theta \vdash \overrightarrow{P}$ and $\Theta \vdash \overrightarrow{P'}$ such that $\Theta \vdash [\overrightarrow{P}/\overrightarrow{\alpha^+}][\overrightarrow{P'}/\overrightarrow{\alpha^{+\prime}}]N' \leqslant M'$ (the decomposition of substitutions is possible since $\overrightarrow{\alpha^+} \cap \Theta = \emptyset$). Then by $(\forall^{\leqslant})$ again, $\Theta \vdash \forall \overrightarrow{\alpha^{+\prime}}. [\overrightarrow{P'}/\overrightarrow{\alpha^{+\prime}}]N' \leqslant M'$ (notice that $[\overrightarrow{P'}/\overrightarrow{\alpha^{+\prime}}]N'$ cannot start with $\forall$).
- $\Leftarrow$ Let us assume $\Theta \vdash [\overrightarrow{P}/\overrightarrow{\alpha^+}]N \leqslant M$ for some $\Theta \vdash \overrightarrow{P}$. let us decompose N as $\forall \overrightarrow{\alpha^{+\prime}}. N'$ where N' does not start with $\forall$. Then $\Theta \vdash [\overrightarrow{P}/\overrightarrow{\alpha^+}]\forall \overrightarrow{\alpha^{+\prime}}. N' \leqslant M'$ or, equivalently, $\Theta \vdash \forall \overrightarrow{\alpha^{+\prime}}. [\overrightarrow{P}/\overrightarrow{\alpha^+}]N' \leqslant M'$ is inferred by $(\forall^{\leqslant})$ (notice that $[\overrightarrow{P}/\overrightarrow{\alpha^+}]N'$ cannot start with $\forall$). By inversion, there exist $\Theta \vdash \overrightarrow{P'}$ such that $\Theta \vdash [\overrightarrow{P'}/\overrightarrow{\alpha^{+\prime}}][\overrightarrow{P}/\overrightarrow{\alpha^+}]N' \leqslant M'$. Since $\overrightarrow{\alpha^{+\prime}}$ is disjoint from the free variables of $\overrightarrow{P}$ and from $\overrightarrow{\alpha^+}$, the composition of $\overrightarrow{P'}/\overrightarrow{\alpha^{+\prime}}$ and $\overrightarrow{P}/\overrightarrow{\alpha^+}$ can be joined into a single substitution well-formed in Θ. Then by $(\forall^{\leqslant})$ again, $\Theta \vdash \forall \overrightarrow{\alpha^+}. N \leqslant M$.

$+$ The positive cases are proved symmetrically.

■

Corollary 7 (Redundant quantifier elimination)**.**

$-_L$ *Suppose that* $\overrightarrow{\alpha^+} \cap \mathtt{fv}(N) = \emptyset$ *then* $\Theta \vdash \forall \overrightarrow{\alpha^+}. N \leqslant M$ *holds if and only if* $\Theta \vdash N \leqslant M$*;*

$-_R$ *Suppose that* $\overrightarrow{\alpha^+} \cap \mathtt{fv}(M) = \emptyset$ *then* $\Theta \vdash N \leqslant \forall \overrightarrow{\alpha^+}. M$ *holds if and only if* $\Theta \vdash N \leqslant M$*;*

$+_L$ *Suppose that* $\overrightarrow{\alpha^-} \cap \mathtt{fv}(P) = \emptyset$ *then* $\Theta \vdash \exists \overrightarrow{\alpha^-}. P \geqslant Q$ *holds if and only if* $\Theta \vdash P \geqslant Q$.

$+_R$ *Suppose that* $\overrightarrow{\alpha^-} \cap \mathtt{fv}(Q) = \emptyset$ *then* $\Theta \vdash P \geqslant \exists \overrightarrow{\alpha^-}. Q$ *holds if and only if* $\Theta \vdash P \geqslant Q$.

Proof

$-_R$ Suppose that $\overrightarrow{\alpha^+} \cap \mathtt{fv}(M) = \emptyset$ then by Lemma 18, $\Theta \vdash N \leqslant \forall\overrightarrow{\alpha^+}.\ M$ is equivalent to $\Theta, \overrightarrow{\alpha^+} \vdash N \leqslant M$, By Lemma 4, since $\overrightarrow{\alpha^+} \cap \mathtt{fv}(N) = \emptyset$ and $\overrightarrow{\alpha^+} \cap \mathtt{fv}(M) = \emptyset$, $\Theta, \overrightarrow{\alpha^+} \vdash N \leqslant M$ is equivalent to $\Theta \vdash N \leqslant M$.

$-_L$ Suppose that $\overrightarrow{\alpha^+} \cap \mathtt{fv}(N) = \emptyset$. Let us decompose M as $\forall\overrightarrow{\beta^+}.\ M'$ where M' does not start with $\forall$. By Lemma 18, $\Theta \vdash \forall\overrightarrow{\alpha^+}.\ N \leqslant \forall\overrightarrow{\beta^+}.\ M'$ is equivalent to $\Theta, \overrightarrow{\beta^+} \vdash \forall\overrightarrow{\alpha^+}.\ N \leqslant M'$, which is equivalent to existence of $\Theta, \overrightarrow{\beta^+} \vdash \overrightarrow{P}$ such that $\Theta, \overrightarrow{\beta^+} \vdash [\overrightarrow{P}/\overrightarrow{\alpha^+}]N \leqslant M'$. Since $[\overrightarrow{P}/\overrightarrow{\alpha^+}]N = N$, the latter is equivalent to $\Theta, \overrightarrow{\beta^+} \vdash N \leqslant M'$, which is equivalent to $\Theta \vdash N \leqslant \forall\overrightarrow{\beta^+}.\ M'$. $\Theta, \overrightarrow{\beta^+} \vdash \overrightarrow{P}$ can be chosen arbitrary, for example, $\overrightarrow{P}_i = \exists\alpha^-.\downarrow\alpha^-$.

$+$ The positive cases are proved symmetrically.

■

Lemma 19 (Subtypes and supertypes of a variable)**.** *Assuming* $\Theta \vdash \alpha^-$, $\Theta \vdash \alpha^+$, $\Theta \vdash N$, *and* $\Theta \vdash P$,

$+$ *if* $\Theta \vdash P \geqslant \exists\overrightarrow{\alpha^-}.\ \alpha^+$ *or* $\Theta \vdash \exists\overrightarrow{\alpha^-}.\ \alpha^+ \geqslant P$ *then* $P = \exists\overrightarrow{\beta^-}.\ \alpha^+$ *(for some potentially empty* $\overrightarrow{\beta^-}$*)*

$-$ *if* $\Theta \vdash N \leqslant \forall\overrightarrow{\alpha^+}.\ \alpha^-$ *or* $\Theta \vdash \forall\overrightarrow{\alpha^+}.\ \alpha^- \leqslant N$ *then* $N = \forall\overrightarrow{\beta^+}.\ \alpha^-$ *(for some potentially empty* $\overrightarrow{\beta^+}$*)*

Proof We prove by induction on the tree inferring $\Theta \vdash P \geqslant \exists\overrightarrow{\alpha^-}.\ \alpha^+$ or $\Theta \vdash \exists\overrightarrow{\alpha^-}.\ \alpha^+ \geqslant P$ or or $\Theta \vdash N \leqslant \forall\overrightarrow{\alpha^+}.\ \alpha^-$ or $\Theta \vdash \forall\overrightarrow{\alpha^+}.\ \alpha^- \leqslant N$.

Let us consider which one of these judgments is inferred.

Case 1. $\Theta \vdash P \geqslant \exists\overrightarrow{\alpha^-}.\ \alpha^+$

If the size of the inference tree is 1 then the only rule that can infer it is $(\textsc{Var}^{\geqslant}_{+})$, which implies that $\overrightarrow{\alpha^-}$ is empty and $P = \alpha^+$.

If the size of the inference tree is > 1 then the last rule inferring it must be $(\exists^{\geqslant})$. By inverting this rule, $P = \exists\overrightarrow{\beta^-}.\ P'$ where P' does not start with $\exists$ and $\Theta, \overrightarrow{\alpha^-} \vdash [\overrightarrow{N}/\overrightarrow{\beta^-}]P' \geqslant \alpha^+$ for some $\Theta, \overrightarrow{\alpha^-} \vdash N_i$.

By the induction hypothesis, $[\overrightarrow{N}/\overrightarrow{\beta^-}]P' = \exists\overrightarrow{\gamma^-}.\ \alpha^+$. What shape can P' have? As mentioned, it does not start with $\exists$, and it cannot start with $\uparrow$ (otherwise, $[\overrightarrow{N}/\overrightarrow{\alpha^-}]P'$ would also start with $\uparrow$ and would not be equal to $\exists\overrightarrow{\beta^-}.\ \alpha^+$). This way, P' is a *positive* variable. As such, $[\overrightarrow{N}/\overrightarrow{\alpha^-}]P' = P'$, and then $P' = \exists\overrightarrow{\gamma^-}.\ \alpha^+$ meaning that $\overrightarrow{\gamma^-}$ is empty and $P' = \alpha^+$. This way, $P = \exists\overrightarrow{\beta^-}.\ P' = \exists\overrightarrow{\beta^-}.\ \alpha^+$, as required.

Case 2. $\Theta \vdash \exists\overrightarrow{\alpha^-}.\ \alpha^+ \geqslant P$

If the size of the inference tree is 1 then the only rule that can infer it is $(\textsc{Var}^{\geqslant}_{+})$, which implies that $\overrightarrow{\alpha^-}$ is empty and $P = \alpha^+$.

If the size of the inference tree is > 1 then the last rule inferring it must be $(\exists^{\geqslant})$. By inverting this rule, $P = \exists\overrightarrow{\beta^-}.\ Q$ where $\Theta, \overrightarrow{\beta^-} \vdash [\overrightarrow{N}/\overrightarrow{\alpha^-}]\alpha^+ \geqslant Q$ and Q does not start with $\exists$. Notice that since α^+ is positive, $[\overrightarrow{N}/\overrightarrow{\alpha^-}]\alpha^+ = \alpha^+$, i.e. $\Theta, \overrightarrow{\beta^-} \vdash \alpha^+ \geqslant Q$.

By the induction hypothesis, $Q = \exists\overrightarrow{\beta^-}'.\ \alpha^+$, and since Q does not start with $\exists$, $\overrightarrow{\beta^-}'$ is empty This way, $P = \exists\overrightarrow{\beta^-}.\ Q = \exists\overrightarrow{\beta^-}.\ \alpha^+$, as required.

Case 3. The negative cases ($\Theta \vdash N \leqslant \forall\overrightarrow{\alpha^+}.\ \alpha^-$ and $\Theta \vdash \forall\overrightarrow{\alpha^+}.\ \alpha^- \leqslant N$) are proved analogously.

■

Corollary 8 (Variables have no proper subtypes and supertypes)**.** *Assuming that all mentioned types are well-formed in* Θ*,*

$$\Theta \vdash P \geqslant \alpha^+ \iff P = \exists\overrightarrow{\beta^-}.\ \alpha^+ \iff \Theta \vdash P \simeq^{\leqslant} \alpha^+ \iff P \simeq^{D} \alpha^+$$
$$\Theta \vdash \alpha^+ \geqslant P \iff P = \exists\overrightarrow{\beta^-}.\ \alpha^+ \iff \Theta \vdash P \simeq^{\leqslant} \alpha^+ \iff P \simeq^{D} \alpha^+$$
$$\Theta \vdash N \leqslant \alpha^- \iff N = \forall\overrightarrow{\beta^+}.\ \alpha^- \iff \Theta \vdash N \simeq^{\leqslant} \alpha^- \iff N \simeq^{D} \alpha^-$$
$$\Theta \vdash \alpha^- \leqslant N \iff N = \forall\overrightarrow{\beta^+}.\ \alpha^- \iff \Theta \vdash N \simeq^{\leqslant} \alpha^- \iff N \simeq^{D} \alpha^-$$

Proof Notice that $\Theta \vdash \exists\overrightarrow{\beta^-}.\ \alpha^+ \simeq^{\leqslant} \alpha^+$ and $\exists\overrightarrow{\beta^-}.\ \alpha^+ \simeq^{D} \alpha^+$ and apply Lemma 19. ■

Lemma 20 (Subtyping context irrelevance)**.** *Suppose that all the mentioned types are well-formed in* Θ_1 *and* Θ_2*. Then*

+ $\Theta_1 \vdash P \geqslant Q$ *is equivalent to* $\Theta_2 \vdash P \geqslant Q$*;*
− $\Theta_1 \vdash N \leqslant M$ *is equivalent to* $\Theta_2 \vdash N \leqslant M$*.*

Proof We prove it by induction on the size of $\Theta_1 \vdash P \geqslant Q$ and mutually, the size of $\Theta_1 \vdash N \leqslant M$.

All the cases except ($\exists^{\geqslant}$) and ($\forall^{\leqslant}$) are proven congruently: first, we apply the inversion to $\Theta_1 \vdash P \geqslant Q$ to obtain the premises of the corresponding rule X, then we apply the induction hypothesis to each premise, and build the inference tree (with Θ_2) by the same rule X.

Suppose that the judgment is inferred by ($\exists^{\geqslant}$). Then we are proving that $\Theta_1 \vdash \exists\overrightarrow{\alpha^-}.\ P \geqslant \exists\overrightarrow{\beta^-}.\ Q$ implies $\Theta_2 \vdash \exists\overrightarrow{\alpha^-}.\ P \geqslant \exists\overrightarrow{\beta^-}.\ Q$ (the other implication is proven symmetrically).

By inversion of $\Theta_1 \vdash \exists\overrightarrow{\alpha^-}.\ P \geqslant \exists\overrightarrow{\beta^-}.\ Q$, we obtain σ such that $\Theta_1, \overrightarrow{\beta^-} \vdash \sigma : \overrightarrow{\alpha^-}$ and $\Theta_1, \overrightarrow{\beta^-} \vdash [\sigma]P \geqslant Q$. By Lemma 17, $\mathtt{fv}\,([\sigma]P) \subseteq \mathtt{fv}\,(Q)$.

From the well-formedness statements $\Theta_i \vdash \exists\overrightarrow{\alpha^-}.\ P$ and $\Theta_i \vdash \exists\overrightarrow{\beta^-}.\ Q$ we have:

- $\Theta_1, \overrightarrow{\alpha^-} \vdash P$, which also means $\Theta_1, \overrightarrow{\beta^-} \vdash [\sigma]P$ by Lemma 5;
- $\Theta_2, \overrightarrow{\alpha^-} \vdash P$;
- $\Theta_1, \overrightarrow{\beta^-} \vdash Q$; and
- $\Theta_2, \overrightarrow{\beta^-} \vdash Q$, which means $\mathtt{fv}\,(Q) \subseteq \Theta_2, \overrightarrow{\beta^-}$ by Lemma 3, and combining it with $\mathtt{fv}\,([\sigma]P) \subseteq \mathtt{fv}\,(Q)$, we have $\mathtt{fv}\,([\sigma]P) \subseteq \Theta_2, \overrightarrow{\beta^-}$.

Let us construct a substitution σ_0 in the following way:

$$\begin{cases} [\sigma_0]\alpha_i^- = [\sigma]\alpha_i^- & \text{for } \alpha_i^- \in \overrightarrow{\alpha^-} \cap \mathtt{fv}\,(P) \\ [\sigma_0]\alpha_i^- = \forall\gamma^+.\uparrow\gamma^+ & \text{for } \alpha_i^- \in \overrightarrow{\alpha^-} \setminus \mathtt{fv}\,(P) \\ [\sigma_0]\gamma^\pm = \gamma^\pm & \text{for any other } \gamma^\pm \end{cases}$$

Notice that

1. $[\sigma_0]P = [\sigma]P$. Since $\sigma_0|_{\mathtt{fv}\,(P)} = \sigma|_{\mathtt{fv}\,(P)}$ as functions (which follows from the construction of σ_0 and the signature of σ), $[\sigma_0]P = [\sigma_0|_{\mathtt{fv}\,(P)}]P = [\sigma|_{\mathtt{fv}\,(P)}]P = [\sigma]P$ (where the first and the last equalities are by Lemma 6).
2. $\mathtt{fv}\,([\sigma]P) \vdash \sigma_0 : \overrightarrow{\alpha^-}$. To show that, let us consider α_i^-
   - if $\alpha_i^- \in \overrightarrow{\alpha^-} \setminus \mathtt{fv}\,(P)$ then $\cdot \vdash [\sigma_0]\alpha_i^-$, which can be weakened to $\mathtt{fv}\,([\sigma]P) \vdash [\sigma_0]\alpha_i^-$;
   - if $\alpha_i^- \in \overrightarrow{\alpha^-} \cap \mathtt{fv}\,(P)$, we have $[\sigma_0]\alpha_i^- = [\sigma]\alpha_i^-$, and thus, by specification of σ, $\Theta_1, \overrightarrow{\beta^+} \vdash [\sigma_0]\alpha_i^-$. By Corollary 1, it means $\mathtt{fv}\,([\sigma_0]\alpha_i^-) \vdash [\sigma_0]\alpha_i^-$, which we weaken (Corollary 2) to $\mathtt{fv}\,([\sigma]P) \vdash [\sigma_0]\alpha_i^-$ (since $\mathtt{fv}\,([\sigma_0]\alpha_i^-) \subseteq \mathtt{fv}\,([\sigma_0]P)$ by Lemma 16, and $[\sigma_0]P = [\sigma]P$, as noted above).

By Corollary 1, $\Theta_1, \overrightarrow{\beta^-} \vdash [\sigma]P$ implies $\mathtt{fv}\,([\sigma]P) \vdash [\sigma]P$, which, since $\mathtt{fv}\,([\sigma]P) \subseteq \Theta_2, \overrightarrow{\beta^-}$, is weakened to $\Theta_2, \overrightarrow{\beta^-} \vdash [\sigma]P$. and rewritten as $\Theta_2, \overrightarrow{\beta^-} \vdash [\sigma_0]P$.

Notice that the premises of the induction hold:

1. $\Theta_i, \overrightarrow{\beta^-} \vdash [\sigma_0]P$,
2. $\Theta_i, \overrightarrow{\beta^-} \vdash Q$, and
3. $\Theta_1, \overrightarrow{\beta^-} \vdash [\sigma_0]P \geqslant Q$, notice that the tree inferring this judgment is the same tree inferring $\Theta_1, \overrightarrow{\beta^-} \vdash [\sigma]P \geqslant Q$ (since $[\sigma_0]P = [\sigma]P$), i.e., it is a subtree of $\Theta_1 \vdash \exists\overrightarrow{\alpha^-}.\, P \geqslant \exists\overrightarrow{\beta^-}.\, Q$.

This way, by the induction hypothesis, $\Theta_2, \overrightarrow{\beta^-} \vdash [\sigma_0]P \geqslant Q$. Combining it with $\Theta_2, \overrightarrow{\beta^-} \vdash \sigma_0 : \overrightarrow{\alpha^-}$ by $(\exists^{\geqslant})$, we obtain $\Theta_2 \vdash \exists\overrightarrow{\alpha^-}.\, P \geqslant \exists\overrightarrow{\beta^-}.\, Q$.

The case of $\Theta_1 \vdash \forall\overrightarrow{\alpha^+}.\, N \leqslant \forall\overrightarrow{\beta^+}.\, M$ is symmetric. ∎

Lemma 21 (Weakening of subtyping context)**.** *Suppose* Θ_1 *and* Θ_2 *are contexts and* $\Theta_1 \subseteq \Theta_2$. *Then*

+ $\Theta_1 \vdash P \geqslant Q$ *implies* $\Theta_2 \vdash P \geqslant Q$;
− $\Theta_1 \vdash N \leqslant M$ *implies* $\Theta_2 \vdash N \leqslant M$.

Proof By straightforward induction on the subtyping derivation. The polymorphic cases follow from Lemma 9. ∎

Lemma 22 (Reflexivity of subtyping)**.** *Assuming all the types are well-formed in* Θ,

− $\Theta \vdash N \leqslant N$
+ $\Theta \vdash P \geqslant P$

Proof Let us prove it by the size of N and mutually, P.

Case 1. $N = \alpha^-$
Then $\Theta \vdash \alpha^- \leqslant \alpha^-$ is inferred immediately by ($\textsc{Var}^{\leqslant}_{-}$).

Case 2. $N = \forall\overrightarrow{\alpha^+}.\, N'$ where $\overrightarrow{\alpha^+}$ is not empty
First, we rename $\overrightarrow{\alpha^+}$ to fresh $\overrightarrow{\beta^+}$ in $\forall\overrightarrow{\alpha^+}.\, N'$ to avoid name clashes: $\forall\overrightarrow{\alpha^+}.\, N' = \forall\overrightarrow{\beta^+}.\, [\overrightarrow{\alpha^+}/\overrightarrow{\beta^+}]N'$. Then to infer $\Theta \vdash \forall\overrightarrow{\alpha^+}.\, N' \leqslant \forall\overrightarrow{\beta^+}.\, [\overrightarrow{\alpha^+}/\overrightarrow{\beta^+}]N'$ we can apply ($\forall^{\leqslant}$), instantiating $\overrightarrow{\alpha^+}$ with $\overrightarrow{\beta^+}$:

- $\mathtt{fv}\, N \cap \overrightarrow{\beta^+} = \emptyset$ by choice of $\overrightarrow{\beta^+}$,
- $\Theta, \overrightarrow{\beta^+} \vdash \beta^+_i$,
- $\Theta, \overrightarrow{\beta^+} \vdash [\overrightarrow{\beta^+}/\overrightarrow{\alpha^+}]N' \leqslant [\overrightarrow{\beta^+}/\overrightarrow{\alpha^+}]N'$ by the induction hypothesis, since the size of $[\overrightarrow{\beta^+}/\overrightarrow{\alpha^+}]N'$ is equal to the size of N', which is smaller than the size of $N = \forall\overrightarrow{\alpha^+}.\, N'$.

Case 3. $N = P \to M$
Then $\Theta \vdash P \to M \leqslant P \to M$ is inferred by ($\to^{\leqslant}$), since $\Theta \vdash P \geqslant P$ and $\Theta \vdash M \leqslant M$ hold the induction hypothesis.

Case 4. $N = \uparrow P$
Then $\Theta \vdash \uparrow P \leqslant \uparrow P$ is inferred by ($\uparrow^{\leqslant}$), since $\Theta \vdash P \geqslant P$ holds by the induction hypothesis.

Case 5. The positive cases are symmetric to the negative ones.

■

Lemma 23 (Substitution preserves subtyipng)**.** *Suppose that all mentioned types are well-formed in Θ_1, and σ is a substitution $\Theta_2 \vdash \sigma : \Theta_1$.*

- *If $\Theta_1 \vdash N \leqslant M$ then $\Theta_2 \vdash [\sigma]N \leqslant [\sigma]M$.*
- *If $\Theta_1 \vdash P \geqslant Q$ then $\Theta_2 \vdash [\sigma]P \geqslant [\sigma]Q$.*

Proof We prove it by induction on the size of the derivation of $\Theta_1 \vdash N \leqslant M$ and mutually, $\Theta_1 \vdash P \geqslant Q$. Let us consider the last rule used in the derivation:

Case 1. ($\textsc{Var}^{\leqslant}_{-}$). Then by inversion, $N = \alpha^-$ and $M = \alpha^-$. By reflexivity of subtyping (Lemma 22), we have $\Theta_2 \vdash [\sigma]\alpha^- \leqslant [\sigma]\alpha^-$, i.e. $\Theta_2 \vdash [\sigma]N \leqslant [\sigma]M$, as required.

Case 2. ($\forall^{\leqslant}$). Then by inversion, $N = \forall\overrightarrow{\alpha^+}.\, N'$, $M = \forall\overrightarrow{\beta^+}.\, M'$, where $\overrightarrow{\alpha^+}$ or $\overrightarrow{\beta^+}$ is not empty. Moreover, $\Theta_1, \overrightarrow{\beta^+} \vdash [\overrightarrow{P}/\overrightarrow{\alpha^+}]N' \leqslant M'$ for some $\Theta_1, \overrightarrow{\beta^+} \vdash \overrightarrow{P}$, and $\mathtt{fv}\, N \cap \overrightarrow{\beta^+} = \emptyset$.

Notice that since the derivation of $\Theta_1, \overrightarrow{\beta^+} \vdash [\overrightarrow{P}/\overrightarrow{\alpha^+}]N' \leqslant M'$ is a subderivation of the derivation of $\Theta \vdash N \leqslant M$, its size is smaller, and hence, the induction hypothesis applies $(\Theta_1, \overrightarrow{\beta^+} \vdash \sigma : \Theta_1, \overrightarrow{\beta^+}$ by Lemma 14$) : \Theta_2, \overrightarrow{\beta^+} \vdash [\sigma][\overrightarrow{P}/\overrightarrow{\alpha^+}]N' \leqslant [\sigma]M'$.

Notice that by convention, $\overrightarrow{\alpha^+}$ and $\overrightarrow{\beta^+}$ are fresh, and thus, $[\sigma]\forall\overrightarrow{\alpha^+}.\, N' = \forall\overrightarrow{\alpha^+}.\, [\sigma]N'$ and $[\sigma]\forall\overrightarrow{\beta^+}.\, M' = \forall\overrightarrow{\beta^+}.\, [\sigma]M'$, which means that the required $\Theta_2, \Theta \vdash [\sigma]\forall\overrightarrow{\alpha^+}.\, N' \leqslant [\sigma]\forall\overrightarrow{\beta^+}.\, M'$ is rewritten as $\Theta_2, \Theta \vdash \forall\overrightarrow{\alpha^+}.\, [\sigma]N' \leqslant \forall\overrightarrow{\beta^+}.\, [\sigma]M'$.

To infer it, we apply ($\forall^{\leqslant}$), instantiating α^+_i with $[\sigma]P_i$:

- $\mathtt{fv}\,[\sigma]N \cap \overrightarrow{\beta^+} = \emptyset$;
- $\Theta_2, \Theta, \overrightarrow{\beta^+} \vdash [\sigma]P_i$, by Lemma 5 since from the inversion, $\Theta_1, \Theta, \overrightarrow{\beta^+} \vdash P_i$;
- $\Theta, \overrightarrow{\beta^+} \vdash [[\sigma]\overrightarrow{P}/\overrightarrow{\alpha^+}][\sigma]N' \leqslant [\sigma]M'$ holds by Lemma 13: Since $\overrightarrow{\alpha^+}$ is fresh, it is disjoint with the domain and the codomain of σ (Θ_1 and Θ_2), and thus, $[\sigma][\overrightarrow{P}/\overrightarrow{\alpha^+}]N' = [\sigma \ll \overrightarrow{P}/\overrightarrow{\alpha^+}][\sigma]N' = [[\sigma]\overrightarrow{P}/\overrightarrow{\alpha^+}][\sigma]N'$. Then $\Theta_2, \Theta, \overrightarrow{\beta^+} \vdash [\sigma][\overrightarrow{P}/\overrightarrow{\alpha^+}]N' \leqslant [\sigma]M'$ holds by the induction hypothesis.

Case 3. $(\to^{\leqslant})$. Then by inversion, $N = P \to N_1$, $M = Q \to M_1$, $\Theta \vdash P \geqslant Q$, and $\Theta \vdash N_1 \leqslant M_1$. And by the induction hypothesis, $\Theta' \vdash [\sigma]P \geqslant [\sigma]Q$ and $\Theta' \vdash [\sigma]N_1 \leqslant [\sigma]M_1$. Then $\Theta' \vdash [\sigma]N \leqslant [\sigma]M$, i.e. $\Theta' \vdash [\sigma]P \to [\sigma]N_1 \leqslant [\sigma]Q \to [\sigma]M_1$, is inferred by $(\to^{\leqslant})$.

Case 4. $(\uparrow^{\leqslant})$. Then by inversion, $N = \uparrow P$, $M = \uparrow Q$, and $\Theta \vdash P \simeq^{\leqslant} Q$, which by inversion means that $\Theta \vdash P \geqslant Q$ and $\Theta \vdash Q \geqslant P$. Then the induction hypothesis applies, and we have $\Theta' \vdash [\sigma]P \geqslant [\sigma]Q$ and $\Theta' \vdash [\sigma]Q \geqslant [\sigma]P$. Then by sequential application of $(\simeq^{\leqslant}_{-})$ and $(\uparrow^{\leqslant})$ to these judgments, we have $\Theta' \vdash \uparrow[\sigma]P \leqslant \uparrow[\sigma]Q$, i.e. $\Theta' \vdash [\sigma]N \leqslant [\sigma]M$, as required.

Case 5. The positive cases are proved symmetrically.

■

Corollary 9 (Substitution preserves subtyping induced equivalence)**.** *Suppose that* $\Theta \vdash \sigma : \Theta_1$. *Then*

+ *if* $\Theta_1 \vdash P$, $\Theta_1 \vdash Q$, *and* $\Theta_1 \vdash P \simeq^{\leqslant} Q$ *then* $\Theta \vdash [\sigma]P \simeq^{\leqslant} [\sigma]Q$
- *if* $\Theta_1 \vdash N$, $\Theta_1 \vdash M$, *and* $\Theta_1 \vdash N \simeq^{\leqslant} M$ *then* $\Theta \vdash [\sigma]N \simeq^{\leqslant} [\sigma]M$

Lemma 24 (Transitivity of subtyping)**.** *Assuming the types are well-formed in* Θ,

- *if* $\Theta \vdash N_1 \leqslant N_2$ *and* $\Theta \vdash N_2 \leqslant N_3$ *then* $\Theta \vdash N_1 \leqslant N_3$,
+ *if* $\Theta \vdash P_1 \geqslant P_2$ *and* $\Theta \vdash P_2 \geqslant P_3$ *then* $\Theta \vdash P_1 \geqslant P_3$.

Proof To prove it, we formulate a stronger property, which will imply the required one, taking $\sigma = \Theta \vdash \mathsf{id} : \Theta$.

Assuming all the types are well-formed in Θ,

- if $\Theta \vdash N \leqslant M_1$, $\Theta \vdash M_2 \leqslant K$, and for $\Theta' \vdash \sigma : \Theta$, $[\sigma]M_1 = [\sigma]M_2$ then $\Theta' \vdash [\sigma]N \leqslant [\sigma]K$
+ if $\Theta \vdash P \geqslant Q_1$, $\Theta \vdash Q_2 \geqslant R$, and for $\Theta' \vdash \sigma : \Theta$, $[\sigma]Q_1 = [\sigma]Q_2$ then $\Theta' \vdash [\sigma]P \geqslant [\sigma]R$

We prove it by induction on $\mathsf{size}(\Theta \vdash N \leqslant M_1) + \mathsf{size}(\Theta \vdash M_2 \leqslant K)$ and mutually, on $\mathsf{size}(\Theta \vdash P \geqslant Q_1) + \mathsf{size}(\Theta \vdash Q_2 \geqslant R)$.

First, let us consider the 3 important cases.

Case 1. Let us consider the case when $M_1 = \forall\overrightarrow{\beta^+}_1.\,\alpha^-$. Then by Lemma 19, $\Theta \vdash N \leqslant M_1$ means that $N = \forall\overrightarrow{\alpha^+}.\,\alpha^-$. $[\sigma]M_1 = [\sigma]M_2$ means that $\forall\overrightarrow{\beta^+}_1.\,[\sigma]\alpha^- = [\sigma]M_2$.

Applying σ to both sides of $\Theta \vdash M_2 \leqslant K$ (by Lemma 23), we obtain $\Theta' \vdash [\sigma]M_2 \leqslant [\sigma]K$, that is $\Theta' \vdash \forall \overrightarrow{\beta^+}_1.\, [\sigma]\alpha^- \leqslant [\sigma]K$. Since $\mathtt{fv}([\sigma]\alpha^-) \subseteq \Theta, \alpha^-$, it is disjoint from $\overrightarrow{\alpha^+}$ and $\overrightarrow{\beta^+}_1$, This way, by Corollary 7, $\Theta' \vdash \forall \overrightarrow{\beta^+}_1.\, [\sigma]\alpha^- \leqslant [\sigma]K$ is equivalent to $\Theta' \vdash [\sigma]\alpha^- \leqslant [\sigma]K$, which is equivalent to $\Theta' \vdash \forall \overrightarrow{\alpha^+}.\, [\sigma]\alpha^- \leqslant [\sigma]K$, that is $\Theta' \vdash [\sigma]N \leqslant [\sigma]K$.

Case 2. Let us consider the case when $M_2 = \forall \overrightarrow{\beta^+}_2.\, \alpha^-$. This case is symmetric to the previous one. Notice that Lemma 19 and Corollary 7 are agnostic to the side on which the quantifiers occur, and thus, the proof stays the same.

Case 3. Let us decompose the types, by extracting the outer quantifiers:

- $N = \forall \overrightarrow{\alpha^+}.\, N'$, where $N' \neq \forall \ldots$,
- $M_1 = \forall \overrightarrow{\beta^+}_1.\, M'_1$, where $M'_1 \neq \forall \ldots$,
- $M_2 = \forall \overrightarrow{\beta^+}_2.\, M'_2$, where $M'_2 \neq \forall \ldots$,
- $K = \forall \overrightarrow{\gamma^+}.\, K'$, where $K' \neq \forall \ldots$.

and assume that at least one of $\overrightarrow{\alpha^+}$, $\overrightarrow{\beta^+}_1$, $\overrightarrow{\beta^+}_2$, and $\overrightarrow{\gamma^+}$ is not empty. Since $[\sigma]M_1 = [\sigma]M_2$, we have $\forall \overrightarrow{\beta^+}_1.\, [\sigma]M'_1 = \forall \overrightarrow{\beta^+}_2.\, [\sigma]M'_2$, and since M'_i are not variables (which was covered by the previous cases) and do not start with $\forall$, $[\sigma]M'_i$ do not start with $\forall$ either, which means $\overrightarrow{\beta^+}_1 = \overrightarrow{\beta^+}_2$ and $[\sigma]M'_1 = [\sigma]M'_2$. Let us rename $\overrightarrow{\beta^+}_1$ and $\overrightarrow{\beta^+}_2$ to $\overrightarrow{\beta^+}$. Then $M_1 = \forall \overrightarrow{\beta^+}.\, M'_1$ and $M_2 = \forall \overrightarrow{\beta^+}.\, M'_2$.

By Lemma 18 applied twice to $\Theta \vdash \forall \overrightarrow{\alpha^+}.\, N' \leqslant \forall \overrightarrow{\beta^+}.\, M'_1$ and to $\Theta \vdash \forall \overrightarrow{\beta^+}.\, M'_2 \leqslant \forall \overrightarrow{\gamma^+}.\, K'$, we have the following:

1. $\Theta, \overrightarrow{\beta^+} \vdash [\overrightarrow{P}/\overrightarrow{\alpha^+}]N' \leqslant M'_1$ for some $\Theta, \overrightarrow{\beta^+} \vdash \overrightarrow{P}$;
2. $\Theta, \overrightarrow{\gamma^+} \vdash [\overrightarrow{Q}/\overrightarrow{\beta^+}]M'_2 \leqslant K'$ for some $\Theta, \overrightarrow{\gamma^+} \vdash \overrightarrow{Q}$.

And since at least one of $\overrightarrow{\alpha^+}$, $\overrightarrow{\beta^+}$, and $\overrightarrow{\gamma^+}$ is not empty, either $\Theta \vdash N \leqslant M_1$ or $\Theta \vdash M_2 \leqslant K$ is inferred by $(\forall^{\leqslant})$, meaning that either $\Theta, \overrightarrow{\beta^+} \vdash [\overrightarrow{P}/\overrightarrow{\alpha^+}]N' \leqslant M'_1$ is a proper subderivation of $\Theta \vdash N \leqslant M_1$ or $\Theta, \overrightarrow{\gamma^+} \vdash [\overrightarrow{Q}/\overrightarrow{\beta^+}]M'_2 \leqslant K'$ is a proper subderivation of $\Theta \vdash M_2 \leqslant K$.

Notice that we can weaken and rearrange the contexts without changing the sizes of the derivations: $\Theta, \overrightarrow{\beta^+}, \overrightarrow{\gamma^+} \vdash [\overrightarrow{P}/\overrightarrow{\alpha^+}]N' \leqslant M'_1$ and $\Theta, \overrightarrow{\beta^+}, \overrightarrow{\gamma^+} \vdash [\overrightarrow{Q}/\overrightarrow{\beta^+}]M'_2 \leqslant K'$. This way, the sum of the sizes of these derivations is smaller than the sum of the sizes of $\Theta \vdash N \leqslant M_1$ and $\Theta \vdash M_2 \leqslant K$. Let us apply the induction hypothesis to these derivations, with the substitution $\Theta', \overrightarrow{\gamma^+} \vdash \sigma \circ (\overrightarrow{Q}/\overrightarrow{\beta^+}) : \Theta, \overrightarrow{\beta^+}, \overrightarrow{\gamma^+}$ (Lemma 14). To apply the induction hypothesis, it is left to show that $\sigma \circ (\overrightarrow{Q}/\overrightarrow{\beta^+})$ unifies M'_1 and $[\overrightarrow{Q}/\overrightarrow{\beta^+}]M'_2$:

$$
\begin{aligned}
[\sigma \circ \overrightarrow{Q}/\overrightarrow{\beta^+}]M'_1 &= [\sigma][\overrightarrow{Q}/\overrightarrow{\beta^+}]M'_1 \\
&= [[\sigma]\overrightarrow{Q}/\overrightarrow{\beta^+}][\sigma]M'_2 && \text{by Lemma 13} \\
&= [[\sigma]\overrightarrow{Q}/\overrightarrow{\beta^+}][\sigma]M'_2 && \text{Since } [\sigma]M'_1 = [\sigma]M'_2
\end{aligned}
$$

$$
\begin{aligned}
&= [\sigma][\overrightarrow{Q}/\overrightarrow{\beta^+}]M'_2 && \text{by Lemma 13}\\
&= [\sigma][\overrightarrow{Q}/\overrightarrow{\beta^+}][\overrightarrow{Q}/\overrightarrow{\beta^+}]M'_2 && \text{Since } \Theta, \overrightarrow{\gamma^+} \vdash \overrightarrow{Q}, \text{ and } (\Theta, \overrightarrow{\gamma^+}) \cap \overrightarrow{\beta^+} = \emptyset\\
&= [\sigma \circ \overrightarrow{Q}/\overrightarrow{\beta^+}][\overrightarrow{Q}/\overrightarrow{\beta^+}]M'_2
\end{aligned}
$$

This way the induction hypothesis gives us $\Theta', \overrightarrow{\gamma^+} \vdash [\sigma][\overrightarrow{Q}/\overrightarrow{\beta^+}][\overrightarrow{P}/\overrightarrow{\alpha^+}]N' \leqslant [\sigma][\overrightarrow{Q}/\overrightarrow{\beta^+}]K'$, and since $\Theta, \overrightarrow{\gamma^+} \vdash K'$, $[\overrightarrow{Q}/\overrightarrow{\beta^+}]K' = K'$, that is $\Theta', \overrightarrow{\gamma^+} \vdash [\sigma][\overrightarrow{Q}/\overrightarrow{\beta^+}][\overrightarrow{P}/\overrightarrow{\alpha^+}]N' \leqslant [\sigma]K'$. Let us rewrite the substitution that we apply to N':

$$
\begin{aligned}
[\sigma \circ \overrightarrow{Q}/\overrightarrow{\beta^+} \circ \overrightarrow{P}/\overrightarrow{\alpha^+}]N' &= [(\sigma \ll \overrightarrow{Q}/\overrightarrow{\beta^+}) \circ \sigma \circ \overrightarrow{P}/\overrightarrow{\alpha^+}]N' && \text{by Lemma 13}\\
&= [(\sigma \ll \overrightarrow{Q}/\overrightarrow{\beta^+}) \circ (\sigma \ll \overrightarrow{P}/\overrightarrow{\alpha^+}) \circ \sigma]N' && \text{by Lemma 13}\\
&= [(((\sigma \ll \overrightarrow{Q}/\overrightarrow{\beta^+}) \circ \sigma) \ll \overrightarrow{P}/\overrightarrow{\alpha^+}) \circ \sigma]N' && \mathtt{fv}([\sigma]N') \cap \overrightarrow{\beta^+} = \emptyset\\
&= [((\sigma \circ \overrightarrow{Q}/\overrightarrow{\beta^+}) \ll \overrightarrow{P}/\overrightarrow{\alpha^+}) \circ \sigma]N' && \text{by Lemma 13}\\
&= [(\sigma \circ \overrightarrow{Q}/\overrightarrow{\beta^+}) \ll \overrightarrow{P}/\overrightarrow{\alpha^+}][\sigma]N'
\end{aligned}
$$

Notice that $(\sigma \circ \overrightarrow{Q}/\overrightarrow{\beta^+}) \ll \overrightarrow{P}/\overrightarrow{\alpha^+}$ is a substitution that turns ${\alpha^+}_i$ into $[\sigma \circ \overrightarrow{Q}/\overrightarrow{\beta^+}]P_i$, where $\Theta', \overrightarrow{\gamma^+} \vdash [\sigma \circ \overrightarrow{Q}/\overrightarrow{\beta^+}]P_i$. This way, $\Theta', \overrightarrow{\gamma^+} \vdash [(\sigma \circ \overrightarrow{Q}/\overrightarrow{\beta^+}) \ll \overrightarrow{P}/\overrightarrow{\alpha^+}][\sigma]N' \leqslant [\sigma]K'$ means $\Theta \vdash \forall\overrightarrow{\alpha^+}.\, [\sigma]N' \leqslant \forall\overrightarrow{\gamma^+}.\, [\sigma]K'$ by Lemma 18, that is $\Theta \vdash [\sigma]N \leqslant [\sigma]K$, as required.

Now, we can assume that neither $\Theta \vdash N \leqslant M_1$ nor $\Theta \vdash M_2 \leqslant K$ is inferred by $(\forall^{\leqslant})$, and that neither M_1 nor M_2 is equivalent to a variable. Because of that, $[\sigma]M_1 = [\sigma]M_2$ means that M_1 and M_2 have the same outer constructor. Let us consider the shape of M_1.

Case 1. $M_1 = \alpha^-$ this case has been considered;

Case 2. $M_1 = \forall\overrightarrow{\beta^+}.\, M'_1$ this case has been considered;

Case 3. $M_1 = \uparrow Q_1$. Then as noted above, $[\sigma]M_1 = [\sigma]M_2$ means that $M_2 = \uparrow Q_2$ and $[\sigma]Q_1 = [\sigma]Q_2$. Moreover, $\Theta \vdash N \leqslant \uparrow Q_1$ can only be inferred by $(\uparrow^{\leqslant})$, and thus, $N = \uparrow P$, and by inversion, $\Theta \vdash P \geqslant Q_1$ and $\Theta \vdash Q_1 \geqslant P$. Analogously, $\Theta \vdash \uparrow Q_2 \leqslant K$ means that $K = \uparrow R$, $\Theta \vdash Q_2 \geqslant R$, and $\Theta \vdash R \geqslant Q_2$.

Notice that the derivations of $\Theta \vdash P \geqslant Q_1$ and $\Theta \vdash Q_1 \geqslant P$ are proper sub-derivations of $\Theta \vdash N \leqslant M_1$, and the derivations of $\Theta \vdash Q_2 \geqslant R$ and $\Theta \vdash R \geqslant Q_2$ are proper sub-derivations of $\Theta \vdash M_2 \leqslant K$. This way, the induction hypothesis is applicable:

- applying the induction hypothesis to $\Theta \vdash P \geqslant Q_1$ and $\Theta \vdash Q_2 \geqslant R$ with $\Theta' \vdash \sigma : \Theta$ unifying Q_1 and Q_2, we obtain $\Theta' \vdash [\sigma]P \geqslant [\sigma]R$;
- applying the induction hypothesis to $\Theta \vdash R \geqslant Q_2$ and $\Theta \vdash Q_1 \geqslant P$ with $\Theta' \vdash \sigma : \Theta$ unifying Q_2 and Q_1, we obtain $\Theta' \vdash [\sigma]R \geqslant [\sigma]P$.

This way, by $(\uparrow^{\leqslant})$, $\Theta' \vdash [\sigma]N \leqslant [\sigma]K$, as required.

Case 4. $M_1 = Q_1 \to M'_1$. Then as noted above, $[\sigma]M_1 = [\sigma]M_2$ means that $M_2 = Q_2 \to M'_2$, $[\sigma]Q_1 = [\sigma]Q_2$, and $[\sigma]M'_1 = [\sigma]M'_2$. Moreover, $\Theta \vdash N \leqslant Q_1 \to M'_1$ can only be inferred by $(\to^{\leqslant})$, and thus, $N = P \to N'$, and by inversion, $\Theta \vdash P \geqslant Q_1$ and

$\Theta \vdash N' \leqslant M'_1$. Analogously, $\Theta \vdash Q_2 \to M'_2 \leqslant K$ means that $K = R \to K'$, $\Theta \vdash Q_2 \geqslant R$, and $\Theta \vdash M'_2 \leqslant K'$.

Notice that the derivations of $\Theta \vdash P \geqslant Q_1$ and $\Theta \vdash N' \leqslant M'_1$ are proper sub-derivations of $\Theta \vdash P \to N' \leqslant Q_1 \to M'_1$, and the derivations of $\Theta \vdash Q_2 \geqslant R$ and $\Theta \vdash M'_2 \leqslant K'$ are proper sub-derivations of $\Theta \vdash Q_2 \to M'_2 \leqslant R \to K'$. This way, the induction hypothesis is applicable:

- applying the induction hypothesis to $\Theta \vdash P \geqslant Q_1$ and $\Theta \vdash Q_2 \geqslant R$ with $\Theta' \vdash \sigma : \Theta$ unifying Q_1 and Q_2, we obtain $\Theta' \vdash [\sigma]P \geqslant [\sigma]R$;
- applying the induction hypothesis to $\Theta \vdash N' \leqslant M'_1$ and $\Theta \vdash M'_2 \leqslant K'$ with $\Theta' \vdash \sigma : \Theta$ unifying M'_1 and M'_2, we obtain $\Theta' \vdash [\sigma]N' \leqslant [\sigma]K'$.

This way, by $(\to^{\leqslant})$, $\Theta' \vdash [\sigma]P \to [\sigma]N' \leqslant [\sigma]R \to [\sigma]K'$, that is $\Theta' \vdash [\sigma]N \leqslant [\sigma]K$, as required.

After that, we consider all the analogous positive cases and prove them symmetrically. ■

Corollary 10 (Transitivity of equivalence)**.** *Assuming the types are well-formed in* Θ*,*

– *if* $\Theta \vdash N_1 \simeq^{\leqslant} N_2$ *and* $\Theta \vdash N_2 \simeq^{\leqslant} N_3$ *then* $\Theta \vdash N_1 \simeq^{\leqslant} N_3$*,*

\+ *if* $\Theta \vdash P_1 \simeq^{\leqslant} P_2$ *and* $\Theta \vdash P_2 \simeq^{\leqslant} P_3$ *then* $\Theta \vdash P_1 \simeq^{\leqslant} P_3$*.*

C.1.4 Equivalence

Lemma 25 (Declarative Equivalence is invariant under bijections)**.** *Suppose* μ *is a bijection* $\mu : vars_1 \leftrightarrow vars_2$*, then*

\+ $P_1 \simeq^D P_2$ *implies* $[\mu]P_1 \simeq^D [\mu]P_2$*, and there exists an inference tree of* $[\mu]P_1 \simeq^D [\mu]P_2$ *with the same shape as the one inferring* $P_1 \simeq^D P_2$*;*

– $N_1 \simeq^D N_2$ *implies* $[\mu]N_1 \simeq^D [\mu]N_2$*, and there exists an inference tree of* $[\mu]N_1 \simeq^D [\mu]N_2$ *with the same shape as the one inferring* $N_1 \simeq^D N_2$*.*

Proof We prove it by induction on $P_1 \simeq^D P_2$ and mutually, on $N_1 \simeq^D N_2$. Let us consider the last rule used in the derivation.

Case 1. $(\forall^{\simeq^D})$

Then we decompose N_1 as $\forall\overrightarrow{\alpha^+}_1.\, M_1$ and N_2 as $\forall\overrightarrow{\alpha^+}_2.\, M_2$, where M_1 and M_2 do not start with $\forall$-quantifiers. where $|\overrightarrow{\alpha^+}_1| + |\overrightarrow{\alpha^+}_2| > 0$. By convention, let us assume that $\overrightarrow{\alpha^+}_1$ and $\overrightarrow{\alpha^+}_2$ are disjoint form $vars_2$ and $vars_1$.

By inversion, $\overrightarrow{\alpha^+}_1 \cap \mathtt{fv}\, M_2 = \emptyset$ and $M_1 \simeq^D [\mu']M_2$ for some bijection $\mu' : (\overrightarrow{\alpha^+}_2 \cap \mathtt{fv}\, M_2) \leftrightarrow (\overrightarrow{\alpha^+}_1 \cap \mathtt{fv}\, M_1)$. Then let us apply the induction hypothesis to $M_1 \simeq^D [\mu']M_2$ to obtain $[\mu]M_1 \simeq^D [\mu][\mu']M_2$ inferred by the tree of the same shape as $M_1 \simeq^D [\mu']M_2$.

Notice that $[\mu]M_1$ and $[\mu]M_2$ do not start with $\forall$, That is $[\mu]\forall\overrightarrow{\alpha^+}_1.\, M_1 \simeq^D [\mu]\forall\overrightarrow{\alpha^+}_2.\, M_2$, rewritten as $\forall\overrightarrow{\alpha^+}_1.\, [\mu]M_1 \simeq^D \forall\overrightarrow{\alpha^+}_2.\, [\mu]M_2$, can be inferred by $(\forall^{\simeq^D})$:

1. $\overrightarrow{\alpha^+}_1$ is disjoint from $vars_2 \cup \mathtt{fv}\, M_2 \subseteq \mathtt{fv}\, [\mu]M_2$;

2. $[\mu]M_1 \simeq^D [\mu'][\mu]M_2$ because $[\mu'][\mu]M_2 = [\mu][\mu']M_2$ (by Corollary 4: $\mu' : (\overrightarrow{\alpha^+}_2 \cap \mathtt{fv}\, M_2) \leftrightarrow (\overrightarrow{\alpha^+}_1 \cap \mathtt{fv}\, M_1)$, $\mu : vars_1 \leftrightarrow vars_2$, $vars_1$ is disjoint from $\overrightarrow{\alpha^+}_2$ and $\overrightarrow{\alpha^+}_1$; $\overrightarrow{\alpha^+}_2$ is disjoint from $vars_1$ and $vars_2$)

Notice that it is the same rule as the one inferring $N_1 \simeq^D N_2$, and thus, the shapes of the trees are the same.

Case 2. $(\textsc{Var}_-^{\simeq^D})$
Then $N_1 = N_2 = \alpha^-$, and the required $[\mu]\alpha^- = [\mu]\alpha^-$ is also inferred by $(\textsc{Var}_-^{\simeq^D})$, since $[\mu]\alpha^-$ is a variable.

Case 3. $(\to^{\simeq^D})$
Then we are proving that $P_1 \to M_1 \simeq^D P_2 \to M_2$ implies $[\mu](P_1 \to M_1) \simeq^D [\mu](P_2 \to M_2)$ (preserving the tree structure).

By inversion, we have $P_1 \simeq^D P_2$ and $M_1 \simeq^D M_2$, and thus, by the induction hypothesis, $[\mu]P_1 \simeq^D [\mu]P_2$ and $[\mu]M_1 \simeq^D [\mu]M_2$. Then $[\mu](P_1 \to M_1) \simeq^D [\mu](P_2 \to M_2)$, or in other words, $[\mu]P_1 \to [\mu]M_1 \simeq^D [\mu]P_2 \to [\mu]M_2$, is inferred by the same rule—$(\to^{\simeq^D})$.

Case 4. $(\uparrow^{\simeq^D})$ This case is done by similar congruent arguments as the previous one.

Case 5. The positive cases are proved symmetrically.

■

Lemma 26. *The set of free variables is invariant under equivalence.*

– *If* $N \simeq^D M$ *then* $\mathtt{fv}\, N = \mathtt{fv}\, M$ *(as sets)*

\+ *If* $P \simeq^D Q$ *then* $\mathtt{fv}\, P = \mathtt{fv}\, Q$ *(as sets)*

Proof Mutual induction on $N \simeq^D M$ and $P \simeq^D Q$ The base cases ($(\textsc{Var}_-^{\simeq^D})$ and $(\textsc{Var}_+^{\simeq^D})$) are trivial. So are $(\uparrow^{\simeq^D})$, $(\downarrow^{\simeq^D})$, and $(\to^{\simeq^D})$, where the required property follows from the induction hypothesis.

Let us consider the case when the equivalence is formed by $(\forall^{\simeq^D})$, that is the equivalence has a shape $\forall\overrightarrow{\alpha^+}.\, N \simeq^D \forall\overrightarrow{\beta^+}.\, M$, and by inversion, there is a bijection $\mu : (\overrightarrow{\beta^+} \cap \mathtt{fv}\, M) \leftrightarrow (\overrightarrow{\alpha^+} \cap \mathtt{fv}\, N)$ such that $N \simeq^D [\mu]M$, which by the induction hypothesis means $\mathtt{fv}\, N = \mathtt{fv}\, [\mu]M = [\mu]\mathtt{fv}\, M$.

Let us ensure by alpha-equivalence that $\overrightarrow{\alpha^+}$ is disjoint from $\mathtt{fv}\, M$. Then $(\mathtt{fv}\, \forall\overrightarrow{\beta^+}.\, M) \setminus \overrightarrow{\alpha^+} = \mathtt{fv}\, \forall\overrightarrow{\beta^+}.\, M$. Then we apply the following chain of equalities: $\mathtt{fv}\, \forall\overrightarrow{\alpha^+}.\, N = \mathtt{fv}\, N \setminus \overrightarrow{\alpha^+} = ([\mu]\mathtt{fv}\, M) \setminus \overrightarrow{\alpha^+} = [\mu](\mathtt{fv}\, \forall\overrightarrow{\beta^+}.\, M \cup (\overrightarrow{\beta^+} \cap \mathtt{fv}\, M)) \setminus \overrightarrow{\alpha^+} = ([\mu]\mathtt{fv}\, \forall\overrightarrow{\beta^+}.\, M \cup [\mu](\overrightarrow{\beta^+} \cap \mathtt{fv}\, M)) \setminus \overrightarrow{\alpha^+} = ([\mu]\mathtt{fv}\, \forall\overrightarrow{\beta^+}.\, M) \setminus \overrightarrow{\alpha^+} = (\mathtt{fv}\, \forall\overrightarrow{\beta^+}.\, M) \setminus \overrightarrow{\alpha^+} = \mathtt{fv}\, \forall\overrightarrow{\beta^+}.\, M$.

Symmetrically, we prove the case when the equivalence is formed by $(\exists^{\simeq^D})$. ■

Lemma 27 (Declarative equivalence is transitive)**.**

\+ *if* $P_1 \simeq^D P_2$ *and* $P_2 \simeq^D P_3$ *then* $P_1 \simeq^D P_3$,

– *if* $N_1 \simeq^D N_2$ *and* $N_2 \simeq^D N_3$ *then* $N_1 \simeq^D N_3$.

Proof We prove it by $\mathsf{size}(P_1 \simeq^D P_2) + \mathsf{size}(P_2 \simeq^D P_3)$ and mutually, $\mathsf{size}(N_1 \simeq^D N_2) + \mathsf{size}(N_2 \simeq^D N_3)$, where by size, we mean the size of the nodes in the corresponding inference tree.

Case 1. First, let us consider the case when either $N_1 \simeq^D N_2$ or $N_2 \simeq^D N_3$ is inferred by $(\forall^{\simeq^D})$. Let us decompose N_1, N_2, and N_3 as follows: $N_1 = \forall\overrightarrow{\alpha^+}_1.\, M_1$, $N_2 = \forall\overrightarrow{\alpha^+}_2.\, M_2$, and $N_3 = \forall\overrightarrow{\alpha^+}_3.\, M_3$.

Then by inversion of $\forall\overrightarrow{\alpha^+}_1.\, M_1 \simeq^D \forall\overrightarrow{\alpha^+}_2.\, M_2$ (or if $\overrightarrow{\alpha^+}_1$ and $\overrightarrow{\alpha^+}_2$ are both empty, by assumption) :

1. $\overrightarrow{\alpha^+}_1 \cap \mathtt{fv}\, M_2 = \emptyset$ and
2. there exists a bijection on variables $\mu_1 : (\overrightarrow{\alpha^+}_2 \cap \mathtt{fv}\, M_2) \leftrightarrow (\overrightarrow{\alpha^+}_1 \cap \mathtt{fv}\, M_1)$ such that $M_1 \simeq^D [\mu_1] M_2$.

Analogously, $\forall\overrightarrow{\alpha^+}_1.\, M_1 \simeq^D \forall\overrightarrow{\alpha^+}_2.\, M_2$ implies:

1. $\overrightarrow{\alpha^+}_2 \cap \mathtt{fv}\, M_3 = \emptyset$ and
2. $M_2 \simeq^D [\mu_2] M_3$ for some bijection $\mu_2 : (\overrightarrow{\alpha^+}_3 \cap \mathtt{fv}\, M_3) \leftrightarrow (\overrightarrow{\alpha^+}_2 \cap \mathtt{fv}\, M_2)$.

Notice that either $M_1 \simeq^D [\mu_1] M_2$ is inferred by a proper sub-tree of $\forall\overrightarrow{\alpha^+}_1.\, M_1 \simeq^D \forall\overrightarrow{\alpha^+}_2.\, M_2$ or $M_2 \simeq^D [\mu_2] M_3$ is inferred by a proper sub-tree of $\forall\overrightarrow{\alpha^+}_2.\, M_2 \simeq^D \forall\overrightarrow{\alpha^+}_3.\, M_3$.

Then by Lemma 25, $[\mu_1] M_2 \simeq^D [\mu_1 \circ \mu_2] M_3$ and moreover, $\mathsf{size}([\mu_1] M_2 \simeq^D [\mu_1 \circ \mu_2] M_3) = \mathsf{size}(M_2 \simeq^D [\mu_2] M_3)$.

Since at least one of the trees inferring $M_1 \simeq^D [\mu_1] M_2$ and $M_2 \simeq^D [\mu_2] M_3$ is a proper sub-tree of the corresponding original tree, $\mathsf{size}(M_1 \simeq^D [\mu_1] M_2) + \mathsf{size}(M_2 \simeq^D [\mu_2] M_3) < \mathsf{size}(\forall\overrightarrow{\alpha^+}_1.\, M_1 \simeq^D \forall\overrightarrow{\alpha^+}_2.\, M_2) + \mathsf{size}(\forall\overrightarrow{\alpha^+}_2.\, M_2 \simeq^D \forall\overrightarrow{\alpha^+}_3.\, M_3)$, i.e., the induction hypothesis is applicable.

By the induction hypothesis, $M_1 \simeq^D [\mu_1 \circ \mu_2] M_3$. Where $\mu_1 \circ \mu_2$ is a bijection on variables $\mu_1 \circ \mu_2 : (\overrightarrow{\alpha^+}_3 \cap \mathtt{fv}\, M_3) \leftrightarrow (\overrightarrow{\alpha^+}_1 \cap \mathtt{fv}\, M_1)$. Then $\forall\overrightarrow{\alpha^+}_1.\, M_1 \simeq^D \forall\overrightarrow{\alpha^+}_3.\, M_3$ by $(\forall^{\simeq^D})$.

Once this case has been considered, we can assume that neither $N_1 \simeq^D N_2$ nor $N_2 \simeq^D N_3$ is inferred by $(\forall^{\simeq^D})$.

Case 2. $N_1 \simeq^D N_2$ is inferred by $(\textsc{Var}_{\simeq^D})$
Then $N_1 = N_2 = \alpha^-$, and thus, $N_1 \simeq^D N_3$ holds since $N_2 \simeq^D N_3$.

Case 3. $N_1 \simeq^D N_2$ is inferred by $(\to^{\simeq^D})$
Then $N_1 = P_1 \to M_1$ and $N_2 = P_2 \to M_2$, and by inversion, $P_1 \simeq^D P_2$ and $M_1 \simeq^D M_2$.

Moreover, since N_3 does not start with $\forall$, $N_2 \simeq^D N_3$ is also inferred by $(\to^{\simeq^D})$, which means that $N_3 = P_3 \to M_3$, $P_2 \simeq^D P_3$, and $M_2 \simeq^D M_3$.

Then by the induction hypothesis, $P_1 \simeq^D P_3$ and $M_1 \simeq^D M_3$, and thus, $P_1 \to M_1 \simeq^D P_3 \to M_3$ by $(\to^{\simeq^D})$.

Case 4. $N_1 \simeq^D N_2$ is inferred by $(\to^{\simeq^D})$

For this case, the reasoning is the same as for the previous one.

Case 5. The positive cases are proved symmetrically.

■

Lemma 28 (Type well-formedness is invariant under equivalence)**.** *Mutual subtyping implies declarative equivalence.*

+ *if* $P \simeq^D Q$ *then* $\Theta \vdash P \iff \Theta \vdash Q$,
− *if* $N \simeq^D M$ *then* $\Theta \vdash N \iff \Theta \vdash M$

Proof We prove it by induction on $P \simeq^D Q$ and mutually, on $N \simeq^D M$. Let us consider the last rule used in the derivation.

Case 1. $(\textsc{Var}_-^{\simeq^D})$, that is $N \simeq^D M$ has shape $\alpha^- \simeq^D \alpha^-$.
Than $\Theta \vdash P \iff \Theta \vdash Q$ is rewritten as $\Theta \vdash \alpha^- \iff \Theta \vdash \alpha^-$, which holds trivially.

Case 2. $(\uparrow^{\simeq^D})$, that is $N \simeq^D M$ has shape $\uparrow P \simeq^D \uparrow Q$.
By inversion, $P \simeq^D Q$, and by the induction hypothesis, $\Theta \vdash P \iff \Theta \vdash Q$. Also notice that $\Theta \vdash \uparrow P \iff \Theta \vdash P$ and $\Theta \vdash \uparrow Q \iff \Theta \vdash Q$ by inversion and $(\uparrow^{\text{WF}})$. This way, $\Theta \vdash \uparrow P \iff \Theta \vdash P \iff \Theta \vdash Q \iff \Theta \vdash \uparrow Q$.

Case 3. $(\to^{\simeq^D})$, that is $N \simeq^D M$ has shape $P \to N' \simeq^D Q \to M'$.
Then by inversion, $P \simeq^D Q$ and $N' \simeq^D M'$, and by the induction hypothesis, $\Theta \vdash P \iff \Theta \vdash Q$ and $\Theta \vdash N' \iff \Theta \vdash M'$.

$$
\begin{array}{lll}
\Theta \vdash P \to N' \iff \Theta \vdash P \text{ and } \Theta \vdash N' & & \text{by inversion and } (\to^{\text{WF}}) \\
\iff \Theta \vdash Q \text{ and } \Theta \vdash M' & & \text{as noted above} \\
\iff \Theta \vdash Q \to M' & & \text{by } (\to^{\text{WF}}) \text{ and inversion}
\end{array}
$$

Case 4. $(\forall^{\simeq^D})$, that is $N \simeq^D M$ has shape $\forall\overrightarrow{\alpha^+}.\, N' \simeq^D \forall\overrightarrow{\beta^+}.\, M'$.
By inversion, $\forall\overrightarrow{\alpha^+}.\, N' \simeq^D \forall\overrightarrow{\beta^+}.\, M'$ means that $\overrightarrow{\alpha^+} \cap \mathtt{fv}\, M = \emptyset$ and that there exists a bijection on variables $\mu : (\overrightarrow{\beta^+} \cap \mathtt{fv}\, M') \leftrightarrow (\overrightarrow{\alpha^+} \cap \mathtt{fv}\, N')$ such that $N' \simeq^D [\mu]M'$.

By inversion and $(\forall^{\text{WF}})$, $\Theta \vdash \forall\overrightarrow{\alpha^+}.\, N'$ is equivalent to $\Theta, \overrightarrow{\alpha^+} \vdash N'$, and by Lemma 4, it is equivalent to $\Theta, (\overrightarrow{\alpha^+} \cap \mathtt{fv}\, N') \vdash N'$, which, by the induction hypothesis, is equivalent to $\Theta, (\overrightarrow{\alpha^+} \cap \mathtt{fv}\, N') \vdash [\mu]M'$.

Analogously, $\Theta \vdash \forall\overrightarrow{\beta^+}.\, M'$ is equivalent to $\Theta, (\overrightarrow{\beta^+} \cap \mathtt{fv}\, M') \vdash M'$. By Lemma 5, it implies $\Theta, (\overrightarrow{\alpha^+} \cap \mathtt{fv}\, M') \vdash [\mu]M'$. And vice versa, $\Theta, (\overrightarrow{\alpha^+} \cap \mathtt{fv}\, M') \vdash [\mu]M'$ implies $\Theta, (\overrightarrow{\beta^+} \cap \mathtt{fv}\, M') \vdash [\mu^{-1}][\mu]M'$.

This way, both $\Theta \vdash \forall\overrightarrow{\alpha^+}.\, N'$ and $\Theta \vdash \forall\overrightarrow{\beta^+}.\, M'$ are equivalent to $\Theta, (\overrightarrow{\alpha^+} \cap \mathtt{fv}\, N') \vdash [\mu]M'$.

Case 5. For the cases of the positive types, the proofs are symmetric.

■

Lemma 29 (Soundness of equivalence)**.** *Declarative equivalence implies mutual subtyping.*

- + *if* $\Theta \vdash P$, $\Theta \vdash Q$, *and* $P \simeq^D Q$ *then* $\Theta \vdash P \simeq^{\leqslant} Q$,
- – *if* $\Theta \vdash N$, $\Theta \vdash M$, *and* $N \simeq^D M$ *then* $\Theta \vdash N \simeq^{\leqslant} M$.

Proof We prove it by mutual induction on $P \simeq^D Q$ and $N \simeq^D M$.

Case 1. $\alpha^- \simeq^D \alpha^-$

Then $\Theta \vdash \alpha^- \leqslant \alpha^-$ by ($\textsc{Var}^{\leqslant}_-$), which immediately implies $\Theta \vdash \alpha^- \simeq^{\leqslant} \alpha^-$ by ($\simeq^{\leqslant}_-$).

Case 2. $\uparrow P \simeq^D \uparrow Q$

Then by inversion of ($\uparrow^{\leqslant}$), $P \simeq^D Q$, and from the induction hypothesis, $\Theta \vdash P \simeq^{\leqslant} Q$, and (by symmetry) $\Theta \vdash Q \simeq^{\leqslant} P$.

When ($\uparrow^{\leqslant}$) is applied to $\Theta \vdash P \simeq^{\leqslant} Q$, it gives us $\Theta \vdash \uparrow P \leqslant \uparrow Q$; when it is applied to $\Theta \vdash Q \simeq^{\leqslant} P$, we obtain $\Theta \vdash \uparrow Q \leqslant \uparrow P$. Together, it implies $\Theta \vdash \uparrow P \simeq^{\leqslant} \uparrow Q$.

Case 3. $P \to N \simeq^D Q \to M$

Then by inversion of ($\to^{\leqslant}$), $P \simeq^D Q$ and $N \simeq^D M$. By the induction hypothesis, $\Theta \vdash P \simeq^{\leqslant} Q$ and $\Theta \vdash N \simeq^{\leqslant} M$, which means by inversion: (i) $\Theta \vdash P \geqslant Q$, (ii) $\Theta \vdash Q \geqslant P$, (iii) $\Theta \vdash N \leqslant M$, (iv) $\Theta \vdash M \leqslant N$. Applying ($\to^{\leqslant}$) to (i) and (iii), we obtain $\Theta \vdash P \to N \leqslant Q \to M$; applying it to (ii) and (iv), we have $\Theta \vdash Q \to M \leqslant P \to N$. Together, it implies $\Theta \vdash P \to N \simeq^{\leqslant} Q \to M$.

Case 4. $\forall \overrightarrow{\alpha^+}.\, N \simeq^D \forall \overrightarrow{\beta^+}.\, M$

Then by inversion, there exists bijection $\mu : (\overrightarrow{\beta^+} \cap \mathtt{fv}\, M) \leftrightarrow (\overrightarrow{\alpha^+} \cap \mathtt{fv}\, N)$, such that $N \simeq^D [\mu]M$. By the induction hypothesis, $\Theta, \overrightarrow{\alpha^+} \vdash N \simeq^{\leqslant} [\mu]M$. From Corollary 9 and the fact that μ is bijective, we also have $\Theta, \overrightarrow{\beta^+} \vdash [\mu^{-1}]N \simeq^{\leqslant} M$.

Let us construct a substitution $\overrightarrow{\alpha^+} \vdash \overrightarrow{P}/\overrightarrow{\beta^+} : \overrightarrow{\beta^+}$ by extending μ with arbitrary positive types on $\overrightarrow{\beta^+} \setminus \mathtt{fv}\, M$.

Notice that $[\mu]M = [\overrightarrow{P}/\overrightarrow{\beta^+}]M$, and therefore, $\Theta, \overrightarrow{\alpha^+} \vdash N \simeq^{\leqslant} [\mu]M$ implies $\Theta, \overrightarrow{\alpha^+} \vdash [\overrightarrow{P}/\overrightarrow{\beta^+}]M \leqslant N$. Then by ($\forall^{\leqslant}$), $\Theta \vdash \forall \overrightarrow{\beta^+}.\, M \leqslant \forall \overrightarrow{\alpha^+}.\, N$.

Analogously, we construct the substitution from μ^{-1}, and use it to instantiate $\overrightarrow{\alpha^+}$ in the application of ($\forall^{\leqslant}$) to infer $\Theta \vdash \forall \overrightarrow{\alpha^+}.\, N \leqslant \forall \overrightarrow{\beta^+}.\, M$.

This way, $\Theta \vdash \forall \overrightarrow{\beta^+}.\, M \leqslant \forall \overrightarrow{\alpha^+}.\, N$ and $\Theta \vdash \forall \overrightarrow{\alpha^+}.\, N \leqslant \forall \overrightarrow{\beta^+}.\, M$ gives us $\Theta \vdash \forall \overrightarrow{\beta^+}.\, M \simeq^{\leqslant} \forall \overrightarrow{\alpha^+}.\, N$.

Case 5. For the cases of the positive types, the proofs are symmetric.

■

Lemma 30 (Subtyping induced by disjoint substitutions)**.** *Suppose that* $\Theta \vdash \sigma_1 : \Theta_1$ *and* $\Theta \vdash \sigma_2 : \Theta_1$, *where* $\Theta_i \subseteq \Theta$ *and* $\Theta_1 \cap \Theta_2 = \emptyset$. *Then*

- – *assuming* $\Theta \vdash N$, $\Theta \vdash [\sigma_1]N \leqslant [\sigma_2]N$ *implies* $\Theta \vdash \sigma_i \simeq^{\leqslant} \mathsf{id} : \mathtt{fv}\, N$
- + *assuming* $\Theta \vdash P$, $\Theta \vdash [\sigma_1]P \geqslant [\sigma_2]P$ *implies* $\Theta \vdash \sigma_i \simeq^{\leqslant} \mathsf{id} : \mathtt{fv}\, P$

Proof Proof by induciton on $\Theta \vdash N$ (and mutually on $\Theta \vdash P$).

Case 1. $N = \alpha^-$

Then $\Theta \vdash [\sigma_1]N \leqslant [\sigma_2]N$ is rewritten as $\Theta \vdash [\sigma_1]\alpha^- \leqslant [\sigma_2]\alpha^-$. Let us consider the following cases:

a. $\alpha^- \notin \Theta_1$ and $\alpha^- \notin \Theta_2$
Then $\Theta \vdash \sigma_i \simeq^{\leqslant} \mathsf{id} : \alpha^-$ holds immediately, since $[\sigma_i]\alpha^- = [\mathsf{id}]\alpha^- = \alpha^-$ and $\Theta \vdash \alpha^- \simeq^{\leqslant} \alpha^-$.

b. $\alpha^- \in \Theta_1$ and $\alpha^- \in \Theta_2$
This case is not possible by assumption: $\Theta_1 \cap \Theta_2 = \emptyset$.

c. $\alpha^- \in \Theta_1$ and $\alpha^- \notin \Theta_2$
Then we have $\Theta \vdash [\sigma_1]\alpha^- \leqslant \alpha^-$, which by Corollary 8 means $\Theta \vdash [\sigma_1]\alpha^- \simeq^{\leqslant} \alpha^-$, and hence, $\Theta \vdash \sigma_1 \simeq^{\leqslant} \mathsf{id} : \alpha^-$.

$\Theta \vdash \sigma_2 \simeq^{\leqslant} \mathsf{id} : \alpha^-$ holds since $[\sigma_2]\alpha^- = \alpha^-$, similarly to case 1.*a*.

d. $\alpha^- \notin \Theta_1$ and $\alpha^- \in \Theta_2$
Then we have $\Theta \vdash \alpha^- \leqslant [\sigma_2]\alpha^-$, which by Corollary 8 means $\Theta \vdash \alpha^- \simeq^{\leqslant} [\sigma_2]\alpha^-$, and hence, $\Theta \vdash \sigma_2 \simeq^{\leqslant} \mathsf{id} : \alpha^-$.

$\Theta \vdash \sigma_1 \simeq^{\leqslant} \mathsf{id} : \alpha^-$ holds since $[\sigma_1]\alpha^- = \alpha^-$, similarly to case 1.*a*.

Case 2. $N = \forall \overrightarrow{\alpha^+}.\, M$

Then by inversion, $\Theta, \overrightarrow{\alpha^+} \vdash M$. $\Theta \vdash [\sigma_1]N \leqslant [\sigma_2]N$ is rewritten as $\Theta \vdash [\sigma_1]\forall \overrightarrow{\alpha^+}.\, M \leqslant [\sigma_2]\forall \overrightarrow{\alpha^+}.\, M$. By the congruence of substitution and by the inversion of $(\forall^{\leqslant})$, $\Theta, \overrightarrow{\alpha^+} \vdash [\overrightarrow{Q}/\overrightarrow{\alpha^+}][\sigma_1]M \leqslant [\sigma_2]M$, where $\Theta, \overrightarrow{\alpha^+} \vdash Q_i$. Let us denote the (Kleisli) composition of σ_1 and $\overrightarrow{Q}/\overrightarrow{\alpha^+}$ as σ_1', noting that $\Theta, \overrightarrow{\alpha^+} \vdash \sigma_1' : \Theta_1, \overrightarrow{\alpha^+}$, and $(\Theta_1, \overrightarrow{\alpha^+}) \cap \Theta_2 = \emptyset$.

Let us apply the induction hypothesis to M and the substitutions σ_1' and σ_2 with $\Theta, \overrightarrow{\alpha^+} \vdash [\sigma_1']M \leqslant [\sigma_2]M$ to obtain:

$$\Theta, \overrightarrow{\alpha^+} \vdash \sigma_1' \simeq^{\leqslant} \mathsf{id} : \mathtt{fv}\, M \tag{C.1}$$

$$\Theta, \overrightarrow{\alpha^+} \vdash \sigma_2 \simeq^{\leqslant} \mathsf{id} : \mathtt{fv}\, M \tag{C.2}$$

Then $\Theta \vdash \sigma_2 \simeq^{\leqslant} \mathsf{id} : \mathtt{fv}\, \forall \overrightarrow{\alpha^+}.\, M$ holds by strengthening of C.2: for any $\beta^\pm \in \mathtt{fv}\, \forall \overrightarrow{\alpha^+}.\, M = \mathtt{fv}\, M \setminus \overrightarrow{\alpha^+}$, $\Theta, \overrightarrow{\alpha^+} \vdash [\sigma_2]\beta^\pm \simeq^{\leqslant} \beta^\pm$ is strengthened to $\Theta \vdash [\sigma_2]\beta^\pm \simeq^{\leqslant} \beta^\pm$, because $\mathtt{fv}\, [\sigma_2]\beta^\pm = \mathtt{fv}\, \beta^\pm = \{\beta^\pm\} \subseteq \Theta$.

To show that $\Theta \vdash \sigma_1 \simeq^{\leqslant} \mathsf{id} : \mathtt{fv}\, \forall \overrightarrow{\alpha^+}.\, M$, let us take an arbitrary $\beta^\pm \in \mathtt{fv}\, \forall \overrightarrow{\alpha^+}.\, M = \mathtt{fv}\, M \setminus \overrightarrow{\alpha^+}$.

$$\begin{aligned}
\beta^\pm &= [\mathsf{id}]\beta^\pm && \text{by definition of } \mathtt{id} \\
&\simeq^{\leqslant} [\sigma_1']\beta^\pm && \text{by C.1} \\
&= [\overrightarrow{Q}/\overrightarrow{\alpha^+}][\sigma_1]\beta^\pm && \text{by definition of } \sigma_1' \\
&= [\sigma_1]\beta^\pm && \text{because } \overrightarrow{\alpha^+} \cap \mathtt{fv}\, [\sigma_1]\beta^\pm \subseteq \overrightarrow{\alpha^+} \cap \Theta = \emptyset
\end{aligned}$$

This way, $\Theta \vdash [\sigma_1]\beta^\pm \simeq^{\leqslant} \beta^\pm$ for any $\beta^\pm \in \mathtt{fv}\, \forall \overrightarrow{\alpha^+}.\, M$ and thus, $\Theta \vdash \sigma_1 \simeq^{\leqslant} \mathsf{id} : \mathtt{fv}\, \forall \overrightarrow{\alpha^+}.\, M$.

Case 3. $N = P \to M$

Then by inversion, $\Theta \vdash P$ and $\Theta \vdash M$. $\Theta \vdash [\sigma_1]N \leqslant [\sigma_2]N$ is rewritten as $\Theta \vdash [\sigma_1](P \to M) \leqslant [\sigma_2](P \to M)$, then by congruence of substitution, $\Theta \vdash [\sigma_1]P \to [\sigma_1]M \leqslant [\sigma_2]P \to [\sigma_2]M$, then by inversion $\Theta \vdash [\sigma_1]P \geqslant [\sigma_2]P$ and $\Theta \vdash [\sigma_1]M \leqslant [\sigma_2]M$.

Applying the induction hypothesis to $\Theta \vdash [\sigma_1]P \geqslant [\sigma_2]P$ and to $\Theta \vdash [\sigma_1]M \leqslant [\sigma_2]M$, we obtain (respectively):

$$\Theta \vdash \sigma_i \simeq^{\leqslant} \mathsf{id} : \mathtt{fv}\ P \tag{C.3}$$

$$\Theta \vdash \sigma_i \simeq^{\leqslant} \mathsf{id} : \mathtt{fv}\ M \tag{C.4}$$

Noting that $\mathtt{fv}\ (P \to M) = \mathtt{fv}\ P \cup \mathtt{fv}\ M$, we combine Eqs. (C.3) and (C.4) to conclude: $\Theta \vdash \sigma_i \simeq^{\leqslant} \mathsf{id} : \mathtt{fv}\ (P \to M)$.

Case 4. $N = \uparrow P$

Then by inversion, $\Theta \vdash P$. $\Theta \vdash [\sigma_1]N \leqslant [\sigma_2]N$ is rewritten as $\Theta \vdash [\sigma_1]\uparrow P \leqslant [\sigma_2]\uparrow P$, then by congruence of substitution and by inversion, $\Theta \vdash [\sigma_1]P \geqslant [\sigma_2]P$

Applying the induction hypothesis to $\Theta \vdash [\sigma_1]P \geqslant [\sigma_2]P$, we obtain $\Theta \vdash \sigma_i \simeq^{\leqslant} \mathsf{id} : \mathtt{fv}\ P$. Since $\mathtt{fv} \uparrow P = \mathtt{fv}\ P$, we can conclude: $\Theta \vdash \sigma_i \simeq^{\leqslant} \mathsf{id} : \mathtt{fv} \uparrow P$.

Case 5. The positive cases are proved symmetrically.

■

Corollary 11 (Substitution cannot induce proper subtypes or supertypes)**.** *Assuming all mentioned types are well-formed in* Θ *and* σ *is a substitution* $\Theta \vdash \sigma : \Theta$*,*

$$\Theta \vdash [\sigma]N \leqslant N \implies \Theta \vdash [\sigma]N \simeq^{\leqslant} N \text{ and } \Theta \vdash \sigma \simeq^{\leqslant} \mathsf{id} : \mathtt{fv}\ N$$

$$\Theta \vdash N \leqslant [\sigma]N \implies \Theta \vdash N \simeq^{\leqslant} [\sigma]N \text{ and } \Theta \vdash \sigma \simeq^{\leqslant} \mathsf{id} : \mathtt{fv}\ N$$

$$\Theta \vdash [\sigma]P \geqslant P \implies \Theta \vdash [\sigma]P \simeq^{\leqslant} P \text{ and } \Theta \vdash \sigma \simeq^{\leqslant} \mathsf{id} : \mathtt{fv}\ P$$

$$\Theta \vdash P \geqslant [\sigma]P \implies \Theta \vdash P \simeq^{\leqslant} [\sigma]P \text{ and } \Theta \vdash \sigma \simeq^{\leqslant} \mathsf{id} : \mathtt{fv}\ P$$

Lemma 31 (Mutual substitution and subtyping)**.** *Assuming that the mentioned types (*P*,* Q*,* N*, and* M*) are well-formed in* Θ *and that the substitutions (*σ_1 *and* σ_2*) have signature* $\Theta \vdash \sigma_i : \Theta$*,*

+ *if* $\Theta \vdash [\sigma_1]P \geqslant Q$ *and* $\Theta \vdash [\sigma_2]Q \geqslant P$
  then there exists a bijection $\mu : \mathtt{fv}\ P \leftrightarrow \mathtt{fv}\ Q$ *such that* $\Theta \vdash \sigma_1 \simeq^{\leqslant} \mu : \mathtt{fv}\ P$ *and* $\Theta \vdash \sigma_2 \simeq^{\leqslant} \mu^{-1} : \mathtt{fv}\ Q$*;*
- *if* $\Theta \vdash [\sigma_1]N \leqslant M$ *and* $\Theta \vdash [\sigma_2]N \leqslant M$
  then there exists a bijection $\mu : \mathtt{fv}\ N \leftrightarrow \mathtt{fv}\ M$ *such that* $\Theta \vdash \sigma_1 \simeq^{\leqslant} \mu : \mathtt{fv}\ N$ *and* $\Theta \vdash \sigma_2 \simeq^{\leqslant} \mu^{-1} : \mathtt{fv}\ M$.

Proof

+ Applying σ_2 to both sides of $\Theta \vdash [\sigma_1]P \geqslant Q$ (by Lemma 23), we have: $\Theta \vdash [\sigma_2 \circ \sigma_1]P \geqslant [\sigma_2]Q$. Composing it with $\Theta \vdash [\sigma_2]Q \geqslant P$ by transitivity (Lemma 24),

we have $\Theta \vdash [\sigma_2 \circ \sigma_1]P \geqslant P$. Then by Corollary 11, $\Theta \vdash \sigma_2 \circ \sigma_1 \simeq^{\leqslant} \mathsf{id} : \mathtt{fv}\, P$. By a symmetric argument, we also have: $\Theta \vdash \sigma_1 \circ \sigma_2 \simeq^{\leqslant} \mathsf{id} : \mathtt{fv}\, Q$.

Now, we prove that $\Theta \vdash \sigma_2 \circ \sigma_1 \simeq^{\leqslant} \mathsf{id} : \mathtt{fv}\, P$ and $\Theta \vdash \sigma_1 \circ \sigma_2 \simeq^{\leqslant} \mathsf{id} : \mathtt{fv}\, Q$ implies that σ_1 and σ_1 are (equivalent to) mutually inverse bijections.

To do so, it suffices to prove that

(i) for any $\alpha^{\pm} \in \mathtt{fv}\, P$ there exists $\beta^{\pm} \in \mathtt{fv}\, Q$ such that $\Theta \vdash [\sigma_1]\alpha^{\pm} \simeq^{\leqslant} \beta^{\pm}$ and $\Theta \vdash [\sigma_2]\beta^{\pm} \simeq^{\leqslant} \alpha^{\pm}$; and

(ii) for any $\beta^{\pm} \in \mathtt{fv}\, Q$ there exists $\alpha^{\pm} \in \mathtt{fv}\, P$ such that $\Theta \vdash [\sigma_2]\beta^{\pm} \simeq^{\leqslant} \alpha^{\pm}$ and $\Theta \vdash [\sigma_1]\alpha^{\pm} \simeq^{\leqslant} \beta^{\pm}$.

Then these correspondences between $\mathtt{fv}\, P$ and $\mathtt{fv}\, Q$ are mutually inverse functions, since for any $\beta^{\pm}$ there can be at most one $\alpha^{\pm}$ such that $\Theta \vdash [\sigma_2]\beta^{\pm} \simeq^{\leqslant} \alpha^{\pm}$ (and vice versa).

(i) Let us take $\alpha^{\pm} \in \mathtt{fv}\, P$.

a. if $\alpha^{\pm}$ is positive ($\alpha^{\pm} = \alpha^{+}$), from $\Theta \vdash [\sigma_2][\sigma_1]\alpha^{+} \simeq^{\leqslant} \alpha^{+}$, by Corollary 8, we have $[\sigma_2][\sigma_1]\alpha^{+} = \exists\overrightarrow{\beta^{-}}.\, \alpha^{+}$.

What shape can $[\sigma_1]\alpha^{+}$ have? It cannot be $\exists\overrightarrow{\alpha^{-}}.\, \downarrow N$ (for potentially empty $\overrightarrow{\alpha^{-}}$), because the outer constructor $\downarrow$ would remain after the substitution σ_2, whereas $\exists\overrightarrow{\beta^{-}}.\, \alpha^{+}$ does not have $\downarrow$. The only case left is $[\sigma_1]\alpha^{+} = \exists\overrightarrow{\alpha^{-}}.\, \gamma^{+}$.

Notice that $\Theta \vdash \exists\overrightarrow{\alpha^{-}}.\, \gamma^{+} \simeq^{\leqslant} \gamma^{+}$, meaning that $\Theta \vdash [\sigma_1]\alpha^{+} \simeq^{\leqslant} \gamma^{+}$. Also notice that $[\sigma_2]\exists\overrightarrow{\alpha^{-}}.\, \gamma^{+} = \exists\overrightarrow{\beta^{-}}.\, \alpha^{+}$ implies $\Theta \vdash [\sigma_2]\gamma^{+} \simeq^{\leqslant} \alpha^{+}$.

b. if $\alpha^{\pm}$ is negative ($\alpha^{\pm} = \alpha^{-}$) from $\Theta \vdash [\sigma_2][\sigma_1]\alpha^{-} \simeq^{\leqslant} \alpha^{-}$, by Corollary 8, we have $[\sigma_2][\sigma_1]\alpha^{-} = \forall\overrightarrow{\beta^{+}}.\, \alpha^{-}$.

What shape can $[\sigma_1]\alpha^{-}$ have? It cannot be $\forall\overrightarrow{\alpha^{+}}.\, \uparrow P$ nor $\forall\overrightarrow{\alpha^{+}}.\, P \to M$ (for potentially empty $\overrightarrow{\alpha^{+}}$), because the outer constructor ($\to$ or $\uparrow$), remaining after the substitution σ_2, is however absent in the resulting $\forall\overrightarrow{\beta^{+}}.\, \alpha^{-}$. Hence, the only case left is $[\sigma_1]\alpha^{-} = \forall\overrightarrow{\alpha^{+}}.\, \gamma^{-}$ Notice that $\Theta \vdash \gamma^{-} \simeq^{\leqslant} \forall\overrightarrow{\alpha^{+}}.\, \gamma^{-}$, meaning that $\Theta \vdash [\sigma_1]\alpha^{-} \simeq^{\leqslant} \gamma^{-}$. Also notice that $[\sigma_2]\forall\overrightarrow{\alpha^{+}}.\, \gamma^{-} = \forall\overrightarrow{\beta^{+}}.\, \alpha^{-}$ implies $\Theta \vdash [\sigma_2]\gamma^{-} \simeq^{\leqslant} \alpha^{-}$.

(ii) The proof is symmetric: We swap P and Q, σ_1 and σ_2, and exploit $\Theta \vdash [\sigma_1][\sigma_2]\alpha^{\pm} \simeq^{\leqslant} \alpha^{\pm}$ instead of $\Theta \vdash [\sigma_2][\sigma_1]\alpha^{\pm} \simeq^{\leqslant} \alpha^{\pm}$.

– The proof is symmetric to the positive case.

■

Lemma 32 (Equivalent substitution act equivalently)**.** *Suppose that* $\Theta' \vdash \sigma_1 : \Theta$ *and* $\Theta' \vdash \sigma_2 : \Theta$ *are substitutions equivalent on their domain, that is* $\Theta' \vdash \sigma_1 \simeq^{\leqslant} \sigma_2 : \Theta$*. Then*

+ *for any* $\Theta \vdash P$, $\Theta' \vdash [\sigma_1]P \simeq^{\leqslant} [\sigma_2]P$*;*

– *for any* $\Theta \vdash N$, $\Theta' \vdash [\sigma_1]N \simeq^{\leqslant} [\sigma_2]N$.

Proof We prove it by induction on P (and mutually on N).

Case 1. $N = \alpha^-$
Then since by inversion, $\alpha^- \in \Theta$, $\Theta' \vdash [\sigma_1]\alpha^- \simeq^{\leqslant} [\sigma_2]\alpha^-$ holds by definition of $\Theta' \vdash \sigma_1 \simeq^{\leqslant} \sigma_2 : \Theta$.

Case 2. $N = \uparrow P$
Then by inversion, $\Theta \vdash P$. By the induction hypothesis, $\Theta' \vdash [\sigma_1]P \simeq^{\leqslant} [\sigma_2]P$, Then by $(\uparrow^{\leqslant})$, $\Theta' \vdash \uparrow[\sigma_1]P \leqslant \uparrow[\sigma_2]P$, and symmetrically, $\Theta' \vdash \uparrow[\sigma_2]P \leqslant \uparrow[\sigma_1]P$, together meaning that $\Theta' \vdash \uparrow[\sigma_1]P \simeq^{\leqslant} \uparrow[\sigma_2]P$, or equivalently, $\Theta' \vdash [\sigma_1]\uparrow P \simeq^{\leqslant} [\sigma_2]\uparrow P$.

Case 3. $N = P \to M$
Then by inversion, $\Theta \vdash P$ and $\Theta \vdash M$. By the induction hypothesis, $\Theta' \vdash [\sigma_1]P \simeq^{\leqslant} [\sigma_2]P$ and $\Theta' \vdash [\sigma_1]M \simeq^{\leqslant} [\sigma_2]M$, that is $\Theta' \vdash [\sigma_1]P \geqslant [\sigma_2]P$, $\Theta' \vdash [\sigma_2]P \geqslant [\sigma_1]P$, $\Theta' \vdash [\sigma_1]M \leqslant [\sigma_2]M$, and $\Theta' \vdash [\sigma_2]M \leqslant [\sigma_1]M$. Then by $(\to^{\leqslant})$, $\Theta' \vdash [\sigma_1]P \to [\sigma_1]M \leqslant [\sigma_2]P \to [\sigma_2]M$, and again by $(\to^{\leqslant})$, $\Theta' \vdash [\sigma_2]P \to [\sigma_2]M \leqslant [\sigma_1]P \to [\sigma_1]M$. This way, $\Theta' \vdash [\sigma_1]P \to [\sigma_1]M \simeq^{\leqslant} [\sigma_2]P \to [\sigma_2]M$, or equivalently, $\Theta' \vdash [\sigma_1](P \to M) \simeq^{\leqslant} [\sigma_2](P \to M)$.

Case 4. $N = \forall\overrightarrow{\alpha^+}.\, M$ We can assume that $\overrightarrow{\alpha^+}$ is disjoint from Θ and Θ'. By inversion, $\Theta \vdash \forall\overrightarrow{\alpha^+}.\, M$ implies $\Theta, \overrightarrow{\alpha^+} \vdash M$. Notice that $\Theta' \vdash \sigma_i : \Theta$ and $\Theta' \vdash \sigma_1 \simeq^{\leqslant} \sigma_2 : \Theta$ can be extended to $\Theta', \overrightarrow{\alpha^+} \vdash \sigma_i : \Theta, \overrightarrow{\alpha^+}$ and $\Theta', \overrightarrow{\alpha^+} \vdash \sigma_1 \simeq^{\leqslant} \sigma_2 : \Theta, \overrightarrow{\alpha^+}$ by Lemma 14. Then by the induction hypothesis, $\Theta', \overrightarrow{\alpha^+} \vdash [\sigma_1]M \simeq^{\leqslant} [\sigma_2]M$, meaning by inversion that $\Theta', \overrightarrow{\alpha^+} \vdash [\sigma_1]M \leqslant [\sigma_2]M$ and $\Theta', \overrightarrow{\alpha^+} \vdash [\sigma_2]M \leqslant [\sigma_1]M$.

To infer $\Theta' \vdash \forall\overrightarrow{\alpha^+}.\, [\sigma_1]M \leqslant \forall\overrightarrow{\alpha^+}.\, [\sigma_2]M$, we apply $(\forall^{\leqslant})$ with the substitution $\Theta', \overrightarrow{\alpha^+} \vdash \mathsf{id} : \overrightarrow{\alpha^+}$, noting that $\Theta', \overrightarrow{\alpha^+} \vdash [\mathsf{id}][\sigma_1]M \leqslant [\sigma_2]M$ holds since $\Theta', \overrightarrow{\alpha^+} \vdash [\sigma_1]M \leqslant [\sigma_2]M$, as noted above.

Symmetrically, we infer $\Theta' \vdash \forall\overrightarrow{\alpha^+}.\, [\sigma_2]M \leqslant \forall\overrightarrow{\alpha^+}.\, [\sigma_1]M$, which together with $\Theta' \vdash \forall\overrightarrow{\alpha^+}.\, [\sigma_1]M \leqslant \forall\overrightarrow{\alpha^+}.\, [\sigma_2]M$ means $\Theta' \vdash \forall\overrightarrow{\alpha^+}.\, [\sigma_1]M \simeq^{\leqslant} \forall\overrightarrow{\alpha^+}.\, [\sigma_2]M$, or equivalently, $\Theta' \vdash [\sigma_1]\forall\overrightarrow{\alpha^+}.\, M \simeq^{\leqslant} [\sigma_2]\forall\overrightarrow{\alpha^+}.\, M$.

Case 5. The positive cases are proved symmetrically.

■

Lemma 33 (Equivalence of polymorphic types)**.**

– *For* $\Theta \vdash \forall\overrightarrow{\alpha^+}.\, N$ *and* $\Theta \vdash \forall\overrightarrow{\beta^+}.\, M$,
if $\Theta \vdash \forall\overrightarrow{\alpha^+}.\, N \simeq^{\leqslant} \forall\overrightarrow{\beta^+}.\, M$ *then there exists a bijection* $\mu : \overrightarrow{\beta^+} \cap \mathtt{fv}\, M \leftrightarrow \overrightarrow{\alpha^+} \cap \mathtt{fv}\, N$ *such that* $\Theta, \overrightarrow{\alpha^+} \vdash N \simeq^{\leqslant} [\mu]M$,

\+ *For* $\Theta \vdash \exists\overrightarrow{\alpha^-}.\, P$ *and* $\Theta \vdash \exists\overrightarrow{\beta^-}.\, Q$,
if $\Theta \vdash \exists\overrightarrow{\alpha^-}.\, P \simeq^{\leqslant} \exists\overrightarrow{\beta^-}.\, Q$ *then there exists a bijection* $\mu : \overrightarrow{\beta^-} \cap \mathtt{fv}\, Q \leftrightarrow \overrightarrow{\alpha^-} \cap \mathtt{fv}\, P$ *such that* $\Theta, \overrightarrow{\beta^-} \vdash P \simeq^{\leqslant} [\mu]Q$.

Proof

– First, by α-conversion, we ensure $\overrightarrow{\alpha^+} \cap \mathtt{fv}\, M = \emptyset$ and $\overrightarrow{\beta^+} \cap \mathtt{fv}\, N = \emptyset$. By inversion, $\Theta \vdash \forall\overrightarrow{\alpha^+}.\, N \simeq^{\leqslant} \forall\overrightarrow{\beta^+}.\, M$ implies

1. $\Theta, \overrightarrow{\beta^{\pm}} \vdash [\sigma_1]N \leqslant M$ for $\Theta, \overrightarrow{\beta^{\pm}} \vdash \sigma_1 : \overrightarrow{\alpha^{\pm}}$ and
2. $\Theta, \overrightarrow{\alpha^{\pm}} \vdash [\sigma_2]M \leqslant N$ for $\Theta, \overrightarrow{\alpha^{\pm}} \vdash \sigma_2 : \overrightarrow{\beta^{\pm}}$.

To apply Lemma 31, we weaken and rearrange the contexts, and extend the substitutions to act as identity outside of their initial domain:

1. $\Theta, \overrightarrow{\alpha^{\pm}}, \overrightarrow{\beta^{\pm}} \vdash [\sigma_1]N \leqslant M$ for $\Theta, \overrightarrow{\alpha^{\pm}}, \overrightarrow{\beta^{\pm}} \vdash \sigma_1 : \Theta, \overrightarrow{\alpha^{\pm}}, \overrightarrow{\beta^{\pm}}$ and
2. $\Theta, \overrightarrow{\alpha^{\pm}}, \overrightarrow{\beta^{\pm}} \vdash [\sigma_2]M \leqslant N$ for $\Theta, \overrightarrow{\alpha^{\pm}}, \overrightarrow{\beta^{\pm}} \vdash \sigma_2 : \Theta, \overrightarrow{\alpha^{\pm}}, \overrightarrow{\beta^{\pm}}$.

Then from Lemma 31, there exists a bijection $\mu : \mathtt{fv}\ M \leftrightarrow \mathtt{fv}\ N$ such that $\Theta, \overrightarrow{\alpha^{\pm}}, \overrightarrow{\beta^{\pm}} \vdash \sigma_2 \simeq^{\leqslant} \mu : \mathtt{fv}\ M$ and $\Theta, \overrightarrow{\alpha^{\pm}}, \overrightarrow{\beta^{\pm}} \vdash \sigma_1 \simeq^{\leqslant} \mu^{-1} : \mathtt{fv}\ N$.

Let us show that $\mu|_{\overrightarrow{\beta^{\pm}}}$ is the appropriate candidate.

First, we show that if we restrict the domain of μ to $\overrightarrow{\beta^{\pm}}$, its range will be contained in $\overrightarrow{\alpha^{\pm}}$.

Let us take $\gamma^+ \in \overrightarrow{\beta^{\pm}} \cap \mathtt{fv}\ M$ and assume $[\mu]\gamma^+ \notin \overrightarrow{\alpha^{\pm}}$. Then since $\Theta, \overrightarrow{\beta^{\pm}} \vdash \sigma_1 : \overrightarrow{\alpha^{\pm}}$, σ_1 acts as identity outside of $\overrightarrow{\alpha^{\pm}}$, i.e. $[\sigma_1][\mu]\gamma^+ = [\mu]\gamma^+$ (notice that γ^+ is in the domain of μ). Since $\Theta, \overrightarrow{\alpha^{\pm}}, \overrightarrow{\beta^{\pm}} \vdash \sigma_1 \simeq^{\leqslant} \mu^{-1} : \mathtt{fv}\ N$, application of σ_1 to $[\mu]\gamma^+ \in \mathtt{fv}\ N$ is equivalent to application of μ^{-1}, then $\Theta, \overrightarrow{\alpha^{\pm}}, \overrightarrow{\beta^{\pm}} \vdash [\mu^{-1}][\mu]\gamma^+ \simeq^{\leqslant} [\mu]\gamma^+$, i.e. $\Theta, \overrightarrow{\alpha^{\pm}}, \overrightarrow{\beta^{\pm}} \vdash \gamma^+ \simeq^{\leqslant} [\mu]\gamma^+$, which means $\gamma^+ \in \mathtt{fv}\ [\mu]\gamma^+ \subseteq \mathtt{fv}\ N$. By assumption, $\gamma^+ \in \overrightarrow{\beta^{\pm}} \cap \mathtt{fv}\ M$, i.e. $\overrightarrow{\beta^{\pm}} \cap \mathtt{fv}\ N \neq \emptyset$, hence contradiction.

Second, we will show $\Theta, \overrightarrow{\alpha^{\pm}} \vdash N \simeq^{\leqslant} [\mu|_{\overrightarrow{\beta^{\pm}}}]M$.

Since $\Theta, \overrightarrow{\alpha^{\pm}} \vdash \sigma_2 : \overrightarrow{\beta^{\pm}}$ and $\Theta, \overrightarrow{\alpha^{\pm}}, \overrightarrow{\beta^{\pm}} \vdash \sigma_2 \simeq^{\leqslant} \mu : \mathtt{fv}\ M$, we have $\Theta, \overrightarrow{\alpha^{\pm}}, \overrightarrow{\beta^{\pm}} \vdash \sigma_2 \simeq^{\leqslant} \mu|_{\overrightarrow{\beta^{\pm}}} : \mathtt{fv}\ M$: for any $\alpha^{\pm} \in \mathtt{fv}\ M \setminus \overrightarrow{\beta^{\pm}}$, $[\sigma_2]\alpha^{\pm} = \alpha^{\pm}$ since $\Theta, \overrightarrow{\alpha^{\pm}} \vdash \sigma_2 : \overrightarrow{\beta^{\pm}}$, and $[\mu|_{\overrightarrow{\beta^{\pm}}}]\alpha^{\pm} = \alpha^{\pm}$ by definition of substitution restriction; for $\beta^+ \in \overrightarrow{\beta^{\pm}}$, $[\mu|_{\overrightarrow{\beta^{\pm}}}]\beta^+ = [\mu]\beta^+$, and thus, $\Theta, \overrightarrow{\alpha^{\pm}}, \overrightarrow{\beta^{\pm}} \vdash [\sigma_2]\beta^+ \simeq^{\leqslant} [\mu]\beta^+$ can be rewritten to $\Theta, \overrightarrow{\alpha^{\pm}}, \overrightarrow{\beta^{\pm}} \vdash [\sigma_2]\beta^+ \simeq^{\leqslant} [\mu|_{\overrightarrow{\beta^{\pm}}}]\beta^+$.

By Lemma 32, $\Theta, \overrightarrow{\alpha^{\pm}}, \overrightarrow{\beta^{\pm}} \vdash \sigma_2 \simeq^{\leqslant} \mu|_{\overrightarrow{\beta^{\pm}}} : \mathtt{fv}\ M$ implies $\Theta, \overrightarrow{\alpha^{\pm}}, \overrightarrow{\beta^{\pm}} \vdash [\sigma_2]M \simeq^{\leqslant} [\mu|_{\overrightarrow{\beta^{\pm}}}]M$. By similar reasoning, $\Theta, \overrightarrow{\alpha^{\pm}}, \overrightarrow{\beta^{\pm}} \vdash [\sigma_1]N \simeq^{\leqslant} [\mu^{-1}|_{\overrightarrow{\alpha^{\pm}}}]N$.

This way, by transitivity of subtyping (Lemma 24),

$$\Theta, \overrightarrow{\alpha^{\pm}}, \overrightarrow{\beta^{\pm}} \vdash [\mu^{-1}|_{\overrightarrow{\alpha^{\pm}}}]N \leqslant M \tag{C.5}$$

$$\Theta, \overrightarrow{\alpha^{\pm}}, \overrightarrow{\beta^{\pm}} \vdash [\mu|_{\overrightarrow{\beta^{\pm}}}]M \leqslant N \tag{C.6}$$

By applying $\mu|_{\overrightarrow{\beta^{\pm}}}$ to both sides of C.5 (Lemma 23), we have $\Theta, \overrightarrow{\alpha^{\pm}}, \overrightarrow{\beta^{\pm}} \vdash [\mu|_{\overrightarrow{\beta^{\pm}}}][\mu^{-1}|_{\overrightarrow{\alpha^{\pm}}}]N \leqslant [\mu|_{\overrightarrow{\beta^{\pm}}}]M$. By contracting $\mu^{-1}|_{\overrightarrow{\alpha^{\pm}}} \circ \mu|_{\overrightarrow{\beta^{\pm}}} = \mu|_{\overrightarrow{\beta^{\pm}}}^{-1} \circ \mu|_{\overrightarrow{\beta^{\pm}}}$ (notice that $\mathtt{fv}\ N \cap \overrightarrow{\beta^{\pm}} = \emptyset$), we have $\Theta, \overrightarrow{\alpha^{\pm}}, \overrightarrow{\beta^{\pm}} \vdash N \leqslant [\mu|_{\overrightarrow{\beta^{\pm}}}]M$, which together with C.6 means $\Theta, \overrightarrow{\alpha^{\pm}}, \overrightarrow{\beta^{\pm}} \vdash N \simeq^{\leqslant} [\mu|_{\overrightarrow{\beta^{\pm}}}]M$, and by strengthening, $\Theta, \overrightarrow{\alpha^{\pm}} \vdash N \simeq^{\leqslant} [\mu|_{\overrightarrow{\beta^{\pm}}}]M$.

\+ The proof is symmetric to the proof of the negative case.

■

Lemma 34 (Completeness of Equivalence)**.** *Mutual subtyping implies declarative equivalence. Assuming all the types below are well-formed in* Θ*:*

- + *if* $\Theta \vdash P \simeq^{\leqslant} Q$ *then* $P \simeq^{D} Q$*,*
- − *if* $\Theta \vdash N \simeq^{\leqslant} M$ *then* $N \simeq^{D} M$*.*

Proof

- − Induction on the sum of sizes of N and M. By inversion, $\Theta \vdash N \simeq^{\leqslant} M$ means $\Theta \vdash N \leqslant M$ and $\Theta \vdash M \leqslant N$. Let us consider the last rule that forms $\Theta \vdash N \leqslant M$:

  Case 1. $(\textsc{Var}^{\leqslant}_{-})$ i.e. $\Theta \vdash N \leqslant M$ is of the form $\Theta \vdash \alpha^{-} \leqslant \alpha^{-}$
  Then $N \simeq^{D} M$ (i.e. $\alpha^{-} \simeq^{D} \alpha^{-}$) holds immediately by $(\textsc{Var}^{\simeq^{D}}_{-})$.

  Case 2. $(\uparrow^{\leqslant})$ i.e. $\Theta \vdash N \leqslant M$ is of the form $\Theta \vdash \uparrow P \leqslant \uparrow Q$
  Then by inversion, $\Theta \vdash P \simeq^{\leqslant} Q$, and by induction hypothesis, $P \simeq^{D} Q$. Then $N \simeq^{D} M$ (i.e. $\uparrow P \simeq^{D} \uparrow Q$) holds by $(\uparrow^{\simeq^{D}})$.

  Case 3. $(\to^{\leqslant})$ i.e. $\Theta \vdash N \leqslant M$ is of the form $\Theta \vdash P \to N' \leqslant Q \to M'$
  Then by inversion, $\Theta \vdash P \geqslant Q$ and $\Theta \vdash N' \leqslant M'$. Notice that $\Theta \vdash M \leqslant N$ is of the form $\Theta \vdash Q \to M' \leqslant P \to N'$, which by inversion means $\Theta \vdash Q \geqslant P$ and $\Theta \vdash M' \leqslant N'$.

  This way, $\Theta \vdash Q \simeq^{\leqslant} P$ and $\Theta \vdash M' \simeq^{\leqslant} N'$. Then by induction hypothesis, $Q \simeq^{D} P$ and $M' \simeq^{D} N'$. Then $N \simeq^{D} M$ (i.e. $P \to N' \simeq^{D} Q \to M'$) holds by $(\to^{\simeq^{D}})$.

  Case 4. $(\forall^{\leqslant})$ i.e. $\Theta \vdash N \leqslant M$ is of the form $\Theta \vdash \forall \overrightarrow{\alpha^{+}}.\, N' \leqslant \forall \overrightarrow{\beta^{+}}.\, M'$
  Then by Lemma 33, $\Theta \vdash \forall \overrightarrow{\alpha^{+}}.\, N' \simeq^{\leqslant} \forall \overrightarrow{\beta^{+}}.\, M'$ means that there exists a bijection $\mu : \overrightarrow{\beta^{+}} \cap \mathtt{fv}\, M' \leftrightarrow \overrightarrow{\alpha^{+}} \cap \mathtt{fv}\, N'$ such that $\Theta, \overrightarrow{\alpha^{+}} \vdash [\mu]M' \simeq^{\leqslant} N'$.

  Notice that the application of bijection μ to M' does not change its size (which is less than the size of M), hence the induction hypothesis applies. This way, $[\mu]M' \simeq^{D} N'$ (and by symmetry, $N' \simeq^{D} [\mu]M'$) holds by induction. Then we apply $(\forall^{\simeq^{D}})$ to get $\forall \overrightarrow{\alpha^{+}}.\, N' \simeq^{D} \forall \overrightarrow{\beta^{+}}.\, M'$, i.e. $N \simeq^{D} M$.

- + The proof is symmetric to the proof of the negative case.

■

C.1.5 Variable Ordering

Observation 2 (Ordering is deterministic)**.** *If* `ord` *vars* `in` $N = \vec{\alpha}_1$ *and* `ord` *vars* `in` $N = \vec{\alpha}_2$ *then* $\vec{\alpha}_1 = \vec{\alpha}_2$*. If* `ord` *vars* `in` $P = \vec{\alpha}_1$ *and* `ord` *vars* `in` $P = \vec{\alpha}_2$ *then* $\vec{\alpha}_1 = \vec{\alpha}_2$*. This way, we can use* `ord` *vars* `in` N *and as a function on* N*, and* `ord` *vars* `in` P *as a function on* P*.*

Proof By mutual structural induction on N and P. Notice that the shape of the term N or P uniquely determines the last used inference rule, and all the premises are deterministic on the input. ■

Lemma 35 (Soundness of variable ordering)**.** *Variable ordering extracts used free variables.*

– $\texttt{ord}\ \mathit{vars}\ \texttt{in}\ N = \mathit{vars} \cap \texttt{fv}\ N$ *(as sets)*

+ $\texttt{ord}\ \mathit{vars}\ \texttt{in}\ P = \mathit{vars} \cap \texttt{fv}\ P$ *(as sets)*

Proof We prove it by mutual induction on $\texttt{ord}\ \mathit{vars}\ \texttt{in}\ N = \vec{\alpha}$ and $\texttt{ord}\ \mathit{vars}\ \texttt{in}\ P = \vec{\alpha}$. The only non-trivial cases are $(\to^{\text{Ord}})$ and $(\forall^{\text{Ord}})$.

Case 1. $(\to^{\text{Ord}})$ Then the inferred ordering judgement has shape $\texttt{ord}\ \mathit{vars}\ \texttt{in}\ P \to N = \vec{\alpha}_1, (\vec{\alpha}_2 \setminus \vec{\alpha}_1)$ and by inversion, $\texttt{ord}\ \mathit{vars}\ \texttt{in}\ P = \vec{\alpha}_1$ and $\texttt{ord}\ \mathit{vars}\ \texttt{in}\ N = \vec{\alpha}_2$.

By definition of free variables, $\mathit{vars} \cap \texttt{fv}\ P \to N = \mathit{vars} \cap \texttt{fv}\ P \cup \mathit{vars} \cap \texttt{fv}\ N$, and since by the induction hypothesis $\mathit{vars} \cap \texttt{fv}\ P = \vec{\alpha}_1$ and $\mathit{vars} \cap \texttt{fv}\ N = \vec{\alpha}_2$, we have $\mathit{vars} \cap \texttt{fv}\ P \to N = \vec{\alpha}_1 \cup \vec{\alpha}_2$.

On the other hand, as a set $\vec{\alpha}_1 \cup \vec{\alpha}_2$ is equal to $\vec{\alpha}_1, (\vec{\alpha}_2 \setminus \vec{\alpha}_1)$.

Case 2. $(\forall^{\text{Ord}})$. Then the inferred ordering judgement has shape $\texttt{ord}\ \mathit{vars}\ \texttt{in}\ \forall\overrightarrow{\alpha^+}.\ N = \vec{\alpha}$, and by inversion, $\mathit{vars} \cap \overrightarrow{\alpha^+} = \emptyset\ \texttt{ord}\ \mathit{vars}\ \texttt{in}\ N = \vec{\alpha}$. The latter implies that $\mathit{vars} \cap \texttt{fv}\ N = \vec{\alpha}$. We need to show that $\mathit{vars} \cap \texttt{fv}\ \forall\overrightarrow{\alpha^+}.\ N = \vec{\alpha}$, or equivalently, that $\mathit{vars} \cap (\texttt{fv}\ N \setminus \overrightarrow{\alpha^+}) = \mathit{vars} \cap \texttt{fv}\ N$, which holds since $\mathit{vars} \cap \overrightarrow{\alpha^+} = \emptyset$.

■

Corollary 12 (Additivity of ordering)**.** *Variable ordering is additive (in terms of set union) with respect to its first argument.*

– $\texttt{ord}\ (\mathit{vars}_1 \cup \mathit{vars}_2)\ \texttt{in}\ N = \texttt{ord}\ \mathit{vars}_1\ \texttt{in}\ N \cup \texttt{ord}\ \mathit{vars}_2\ \texttt{in}\ N$ *(as sets)*

+ $\texttt{ord}\ (\mathit{vars}_1 \cup \mathit{vars}_2)\ \texttt{in}\ P = \texttt{ord}\ \mathit{vars}_1\ \texttt{in}\ P \cup \texttt{ord}\ \mathit{vars}_2\ \texttt{in}\ P$ *(as sets)*

Lemma 36 (Weakening of ordering)**.** *Only used variables matter in the first argument of the ordering,*

– $\texttt{ord}\ (\mathit{vars} \cap \texttt{fv}\ N)\ \texttt{in}\ N = \texttt{ord}\ \mathit{vars}\ \texttt{in}\ N$

+ $\texttt{ord}\ (\mathit{vars} \cap \texttt{fv}\ P)\ \texttt{in}\ P = \texttt{ord}\ \mathit{vars}\ \texttt{in}\ P$

Proof Mutual structural induction on N and P.

Case 1. If N is a variable α^-, we notice that $\alpha^- \in \mathit{vars}$ is equivalent to $\alpha^- \in \mathit{vars} \cap \alpha^-$.

Case 2. If N has shape $\uparrow P$, then the required property holds immediately by the induction hypothesis, since $\texttt{fv}\ (\uparrow P) = \texttt{fv}\ (P)$.

Case 3. If the term has shape $P \to N$ then $(\to^{\text{Ord}})$ was applied to infer $\texttt{ord}\ (\mathit{vars} \cap (\texttt{fv}\ P \cup \texttt{fv}\ N))\ \texttt{in}\ P \to N$ and $\texttt{ord}\ \mathit{vars}\ \texttt{in}\ P \to N$. By inversion, the result of $\texttt{ord}\ (\mathit{vars} \cap (\texttt{fv}\ P \cup \texttt{fv}\ N))\ \texttt{in}\ P \to N$ depends on $A = \texttt{ord}\ (\mathit{vars} \cap (\texttt{fv}\ P \cup \texttt{fv}\ N))\ \texttt{in}\ P$ and $B = \texttt{ord}\ (\mathit{vars} \cap (\texttt{fv}\ P \cup \texttt{fv}\ N))\ \texttt{in}\ N$. The result of $\texttt{ord}\ \mathit{vars}\ \texttt{in}\ P \to N$ depends on $X = \texttt{ord}\ \mathit{vars}\ \texttt{in}\ P$ and $Y = \texttt{ord}\ \mathit{vars}\ \texttt{in}\ N$.

Let us show that $A = B$ and $X = Y$, so the results are equal. By the induction hypothesis and set properties, $\texttt{ord}\ (\mathit{vars} \cap (\texttt{fv}\ P \cup \texttt{fv}\ N))\ \texttt{in}\ P = \texttt{ord}\ (\mathit{vars} \cap (\texttt{fv}\ P \cup \texttt{fv}\ N)) \cap \texttt{fv}\ (P)\ \texttt{in}\ P = \texttt{ord}\ \mathit{vars} \cap \texttt{fv}\ (P)\ \texttt{in}\ P = \texttt{ord}\ \mathit{vars}\ \texttt{in}\ P$. Analogously, $\texttt{ord}\ (\mathit{vars} \cap (\texttt{fv}\ P \cup \texttt{fv}\ N))\ \texttt{in}\ N = \texttt{ord}\ \mathit{vars}\ \texttt{in}\ N$.

Case 4. If the term has shape $\forall\overrightarrow{\alpha^+}.\, N$, we can assume that $\overrightarrow{\alpha^+}$ is disjoint from *vars*, since we operate on alpha-equivalence classes. Then using the induction hypothesis, set properties and $(\forall^{\text{Ord}})$: $\texttt{ord}\, vars \cap (\texttt{fv}\, (\forall\overrightarrow{\alpha^+}.\, N))\ \texttt{in}\ \forall\overrightarrow{\alpha^+}.\, N = \texttt{ord}\, vars \cap (\texttt{fv}\, (N) \setminus \overrightarrow{\alpha^+})\ \texttt{in}\ N = \texttt{ord}\, vars \cap (\texttt{fv}\, (N) \setminus \overrightarrow{\alpha^+}) \cap \texttt{fv}\, (N)\ \texttt{in}\ N = \texttt{ord}\, vars \cap \texttt{fv}\, (N)\ \texttt{in}\ N = \texttt{ord}\, vars\ \texttt{in}\ N$.

■

Corollary 13 (Idempotency of ordering)**.**

– *If* $\texttt{ord}\, vars\ \texttt{in}\ N = \vec{\alpha}$ *then* $\texttt{ord}\, \vec{\alpha}\ \texttt{in}\ N = \vec{\alpha}$,

\+ *If* $\texttt{ord}\, vars\ \texttt{in}\ P = \vec{\alpha}$ *then* $\texttt{ord}\, \vec{\alpha}\ \texttt{in}\ P = \vec{\alpha}$;

Proof By Lemmas 35 and 36. ■

Next, we make a set-theoretical observation that will be useful further. In general, any injective function (its image) distributes over the set intersection. However, for convenience, we allow the bijections on variables to be applied *outside of their domains* (as identities), which may violate the injectivity. To deal with these cases, we define a special notion of bijections collision-free on certain sets in such a way that a bijection that is collision-free on P and Q, distributes over intersection of P and Q.

Definition 29 (Collision-free bijection)**.** *We say that a bijection* $\mu : A \leftrightarrow B$ *between sets of variables is* ***collision-free on sets*** P *and* Q *if and only if*

1. $\mu(P \cap A) \cap Q = \emptyset$
2. $\mu(Q \cap A) \cap P = \emptyset$

Observation 19. *Suppose that* $\mu : A \leftrightarrow B$ *is a bijection between two sets of variables, and* μ *is collision-free on* P *and* Q*. Then* $\mu(P \cap Q) = \mu(P) \cap \mu(Q)$.

Lemma 37 (Distributivity of renaming over variable ordering)**.** *Suppose that* μ *is a bijection between two sets of variables* $\mu : A \leftrightarrow B$.

– *If* μ *is collision-free on* *vars* *and* $\texttt{fv}\, N$ *then* $[\mu](\texttt{ord}\, vars\ \texttt{in}\ N) = \texttt{ord}\, ([\mu]vars)\ \texttt{in}\ [\mu]N$

\+ *If* μ *is collision-free on* *vars* *and* $\texttt{fv}\, P$ *then* $[\mu](\texttt{ord}\, vars\ \texttt{in}\ P) = \texttt{ord}\, ([\mu]vars)\ \texttt{in}\ [\mu]P$

Proof Mutual induction on N and P.

Case 1. $N = \alpha^-$

let us consider four cases:

a. $\alpha^- \in A$ and $\alpha^- \in vars$. Then

$$
\begin{aligned}
[\mu](\texttt{ord}\, vars\ \texttt{in}\ N) &= [\mu](\texttt{ord}\, vars\ \texttt{in}\ \alpha^-) \\
&= [\mu]\alpha^- && \text{by } (\text{VAR}^{\text{Ord}}_{+\in}) \\
&= \beta^- && \text{for some } \beta^- \in B \text{ (notice } \beta^- \in [\mu]vars) \\
&= \texttt{ord}\, [\mu]vars\ \texttt{in}\ \beta^- && \text{by } (\text{VAR}^{\text{Ord}}_{+\in}), \text{ because } \beta^- \in [\mu]vars \\
&= \texttt{ord}\, [\mu]vars\ \texttt{in}\ [\mu]\alpha^-
\end{aligned}
$$

b. $\alpha^- \notin A$ and $\alpha^- \notin vars$
Notice that $[\mu](\mathtt{ord}\ vars\ \mathtt{in}\ N) = [\mu](\mathtt{ord}\ vars\ \mathtt{in}\ \alpha^-) = \cdot$ by $(\mathrm{VAR}^{\mathrm{ORD}}_{+\notin})$. On the other hand, $\mathtt{ord}\ [\mu]vars\ \mathtt{in}\ [\mu]\alpha^- = \mathtt{ord}\ [\mu]vars\ \mathtt{in}\ \alpha^- = \cdot$ The latter equality is from $(\mathrm{VAR}^{\mathrm{ORD}}_{+\notin})$, because μ is collision-free on $vars$ and $\mathtt{fv}\ N$, so $\mathtt{fv}\ N \ni \alpha^- \notin \mu(A \cap vars) \cup vars \supseteq [\mu]vars$.

c. $\alpha^- \in A$ but $\alpha^- \notin vars$
Then $[\mu](\mathtt{ord}\ vars\ \mathtt{in}\ N) = [\mu](\mathtt{ord}\ vars\ \mathtt{in}\ \alpha^-) = \cdot$ by $(\mathrm{VAR}^{\mathrm{ORD}}_{+\notin})$. To prove that $\mathtt{ord}\ [\mu]vars\ \mathtt{in}\ [\mu]\alpha^- = \cdot$, we apply $(\mathrm{VAR}^{\mathrm{ORD}}_{+\notin})$. Let us show that $[\mu]\alpha^- \notin [\mu]vars$. Since $[\mu]\alpha^- = \mu(\alpha^-)$ and $[\mu]vars \subseteq \mu(A \cap vars) \cup vars$, it suffices to prove $\mu(\alpha^-) \notin \mu(A \cap vars) \cup vars$.

(i) If there is an element $x \in A \cap vars$ such that $\mu x = \mu\alpha^-$, then $x = \alpha^-$ by bijectivity of μ, which contradicts with $\alpha^- \notin vars$. This way, $\mu(\alpha^-) \notin \mu(A \cap vars)$.

(ii) Since μ is collision-free on $vars$ and $\mathtt{fv}\ N$, $\mu(A \cap \mathtt{fv}\ N) \ni \mu(\alpha^-) \notin vars$.

d. $\alpha^- \notin A$ but $\alpha^- \in vars$
$\mathtt{ord}\ [\mu]vars\ \mathtt{in}\ [\mu]\alpha^- = \mathtt{ord}\ [\mu]vars\ \mathtt{in}\ \alpha^- = \alpha^-$. The latter is by $(\mathrm{VAR}^{\mathrm{ORD}}_{+\notin})$, because $\alpha^- = [\mu]\alpha^- \in [\mu]vars$ since $\alpha^- \in vars$. On the other hand, $[\mu](\mathtt{ord}\ vars\ \mathtt{in}\ N) = [\mu](\mathtt{ord}\ vars\ \mathtt{in}\ \alpha^-) = [\mu]\alpha^- = \alpha^-$.

Case 2. $N = \uparrow P$

$$
\begin{aligned}
[\mu](\mathtt{ord}\ vars\ \mathtt{in}\ N) &= [\mu](\mathtt{ord}\ vars\ \mathtt{in}\ \uparrow P) \\
&= [\mu](\mathtt{ord}\ vars\ \mathtt{in}\ P) && \text{by } (\uparrow^{\mathrm{ORD}}) \\
&= \mathtt{ord}\ [\mu]vars\ \mathtt{in}\ [\mu]P && \text{by the induction hypothesis} \\
&= \mathtt{ord}\ [\mu]vars\ \mathtt{in}\ \uparrow[\mu]P && \text{by } (\uparrow^{\mathrm{ORD}}) \\
&= \mathtt{ord}\ [\mu]vars\ \mathtt{in}\ [\mu]\uparrow P && \text{by the definition of substitution} \\
&= \mathtt{ord}\ [\mu]vars\ \mathtt{in}\ [\mu]N
\end{aligned}
$$

Case 3. $N = P \to M$

$$
\begin{aligned}
&[\mu](\mathtt{ord}\ vars\ \mathtt{in}\ N) \\
&= [\mu](\mathtt{ord}\ vars\ \mathtt{in}\ P \to M) \\
&= [\mu](\vec{\alpha}_1, (\vec{\alpha}_2 \setminus \vec{\alpha}_1)) && \text{where } \mathtt{ord}\ vars\ \mathtt{in}\ P = \vec{\alpha}_1 \text{ and } \mathtt{ord}\ vars\ \mathtt{in}\ M = \vec{\alpha}_2 \\
&= [\mu]\vec{\alpha}_1, [\mu](\vec{\alpha}_2 \setminus \vec{\alpha}_1) \\
&= [\mu]\vec{\alpha}_1, ([\mu]\vec{\alpha}_2 \setminus [\mu]\vec{\alpha}_1) && \text{by induction on } \vec{\alpha}_2\text{; the inductive step is similar to case 1.} \\
& && \text{Notice that } \mu \text{ is collision free on } \vec{\alpha}_1 \text{ and } \vec{\alpha}_2 \\
& && \text{since } \vec{\alpha}_1 \subseteq vars \text{ and } \vec{\alpha}_2 \subseteq \mathtt{fv}\ N \\
&= [\mu]\vec{\alpha}_1, ([\mu]\vec{\alpha}_2 \setminus [\mu]\vec{\alpha}_1)
\end{aligned}
$$

On the other hand, $\mathtt{ord}\ [\mu]vars\ \mathtt{in}\ [\mu]N = \mathtt{ord}\ [\mu]vars\ \mathtt{in}\ [\mu]P \to [\mu]M = \vec{\beta}_1, (\vec{\beta}_2 \setminus \vec{\beta}_1) = [\mu]\vec{\alpha}_1, ([\mu]\vec{\alpha}_2 \setminus [\mu]\vec{\alpha}_1)$, where $\mathtt{ord}\ [\mu]vars\ \mathtt{in}\ [\mu]P = \vec{\beta}_1$ and

$\texttt{ord}\,[\mu]vars\,\texttt{in}\,[\mu]M = \overrightarrow{\beta}_2$, then by the induction hypothesis, $\overrightarrow{\beta}_1 = [\mu]\overrightarrow{\alpha}_1$, $\overrightarrow{\beta}_2 = [\mu]\overrightarrow{\alpha}_2$.

Case 4. $N = \forall\overrightarrow{\alpha^+}.\, M$

$$\begin{aligned}[\mu](\texttt{ord}\,vars\,\texttt{in}\,N) &= [\mu]\texttt{ord}\,vars\,\texttt{in}\,\forall\overrightarrow{\alpha^+}.\, M\\ &= [\mu]\texttt{ord}\,vars\,\texttt{in}\,M\\ &= \texttt{ord}\,[\mu]vars\,\texttt{in}\,[\mu]M \qquad \text{by the induction hypothesis}\end{aligned}$$

$$\begin{aligned}(\texttt{ord}\,[\mu]vars\,\texttt{in}\,[\mu]N) &= \texttt{ord}\,[\mu]vars\,\texttt{in}\,[\mu]\forall\overrightarrow{\alpha^+}.\, M\\ &= \texttt{ord}\,[\mu]vars\,\texttt{in}\,\forall\overrightarrow{\alpha^+}.\,[\mu]M\\ &= \texttt{ord}\,[\mu]vars\,\texttt{in}\,[\mu]M\end{aligned}$$

■

Lemma 38 (Ordering is not affected by independent substitutions)**.** *Suppose that* $\Theta_2 \vdash \sigma : \Theta_1$*, i.e.* σ *maps variables from* Θ_1 *into types taking free variables from* Θ_2*, and vars is a set of variables disjoint with both* Θ_1 *and* Θ_2*,* N *and* P *are types. Then*

- $\texttt{ord}\,vars\,\texttt{in}\,[\sigma]N = \texttt{ord}\,vars\,\texttt{in}\,N$
- $\texttt{ord}\,vars\,\texttt{in}\,[\sigma]P = \texttt{ord}\,vars\,\texttt{in}\,P$

Proof Mutual induction on N and P.

Case 1. $N = \alpha^-$

If $\alpha^- \notin \Theta_1$ then $[\sigma]\alpha^- = \alpha^-$ and $\texttt{ord}\,vars\,\texttt{in}\,[\sigma]\alpha^- = \texttt{ord}\,vars\,\texttt{in}\,\alpha^-$, as requried. If $\alpha^- \in \Theta_1$ then $\alpha^- \notin vars$, so $\texttt{ord}\,vars\,\texttt{in}\,\alpha^- = \cdot$. Moreover, $\Theta_2 \vdash \sigma : \Theta_1$ means $\texttt{fv}\,([\sigma]\alpha^-) \subseteq \Theta_2$, and thus, as a set, $\texttt{ord}\,vars\,\texttt{in}\,[\sigma]\alpha^- = vars \cap \texttt{fv}\,([\sigma]\alpha^-) \subseteq vars \cap \Theta_2 = \cdot$.

Case 2. $N = \forall\overrightarrow{\alpha^+}.\, M$ We can assume $\overrightarrow{\alpha^+} \cap \Theta_1 = \emptyset$ and $\overrightarrow{\alpha^+} \cap vars = \emptyset$. Then

$$\begin{aligned}[t]\texttt{ord}\,vars\,\texttt{in}\,[\sigma]N &= \texttt{ord}\,vars\,\texttt{in}\,[\sigma]\forall\overrightarrow{\alpha^+}.\, M\\ &= \texttt{ord}\,vars\,\texttt{in}\,\forall\overrightarrow{\alpha^+}.\,[\sigma]M\\ &= \texttt{ord}\,vars\,\texttt{in}\,[\sigma]M \qquad \text{by the induction hypothesis}\\ &= \texttt{ord}\,vars\,\texttt{in}\,M\\ &= \texttt{ord}\,vars\,\texttt{in}\,\forall\overrightarrow{\alpha^+}.\, M\\ &= \texttt{ord}\,vars\,\texttt{in}\,N\end{aligned}$$

Case 3. $N = \uparrow P$

$$\begin{aligned}[t]\texttt{ord}\,vars\,\texttt{in}\,[\sigma]N &= \texttt{ord}\,vars\,\texttt{in}\,[\sigma]\uparrow P\\ &= \texttt{ord}\,vars\,\texttt{in}\,\uparrow[\sigma]P \qquad \text{by the definition of substitution}\\ &= \texttt{ord}\,vars\,\texttt{in}\,[\sigma]P \qquad \text{by the induction hypothesis}\end{aligned}$$

$= \mathtt{ord}\ vars\ \mathtt{in}\ P$ by the definition of substitution

$= \mathtt{ord}\ vars\ \mathtt{in} \uparrow P$ by the definition of ordering

$= \mathtt{ord}\ vars\ \mathtt{in}\ N$

Case 4. $N = P \to M$

$\mathtt{ord}\ vars\ \mathtt{in}\ [\sigma]N = \mathtt{ord}\ vars\ \mathtt{in}\ [\sigma](P \to M)$

$= \mathtt{ord}\ vars\ \mathtt{in}\ ([\sigma]P \to [\sigma]M)$ def. of substitution

$= \mathtt{ord}\ vars\ \mathtt{in}\ [\sigma]P,$

$(\mathtt{ord}\ vars\ \mathtt{in}\ [\sigma]M \setminus \mathtt{ord}\ vars\ \mathtt{in}\ [\sigma]P)$ def. of ordering

$= \mathtt{ord}\ vars\ \mathtt{in}\ P,$

$(\mathtt{ord}\ vars\ \mathtt{in}\ M \setminus \mathtt{ord}\ vars\ \mathtt{in}\ P)$ the induction hypothesis

$= \mathtt{ord}\ vars\ \mathtt{in}\ P \to M$ def. of ordering

$= \mathtt{ord}\ vars\ \mathtt{in}\ N$

Case 5. The proofs of the positive cases are symmetric.

■

Lemma 39 (Completeness of variable ordering)**.** *Variable ordering is invariant under equivalence. For arbitrary vars,*

– *If* $N \simeq^D M$ *then* $\mathtt{ord}\ vars\ \mathtt{in}\ N = \mathtt{ord}\ vars\ \mathtt{in}\ M$ *(as lists)*

\+ *If* $P \simeq^D Q$ *then* $\mathtt{ord}\ vars\ \mathtt{in}\ P = \mathtt{ord}\ vars\ \mathtt{in}\ Q$ *(as lists)*

Proof Mutual induction on $N \simeq^D M$ and $P \simeq^D Q$. Let us consider the rule inferring $N \simeq^D M$.

Case 1. $(\textsc{Var}_{-}^{\simeq^D})$

Case 2. $(\uparrow^{\simeq^D})$

Case 3. $(\to^{\simeq^D})$ Then the equivalence has shape $P \to N \simeq^D Q \to M$, and by inversion, $P \simeq^D Q$ and $N \simeq^D M$. Then by the induction hypothesis, $\mathtt{ord}\ vars\ \mathtt{in}\ P = \mathtt{ord}\ vars\ \mathtt{in}\ Q$ and $\mathtt{ord}\ vars\ \mathtt{in}\ N = \mathtt{ord}\ vars\ \mathtt{in}\ M$. Since the resulting ordering for $P \to N$ and $Q \to M$ depend on the ordering of the corresponding components, which are equal, the results are equal.

Case 4. $(\forall^{\simeq^D})$ Then the equivalence has shape $\forall\overrightarrow{\alpha^+}.\ N \simeq^D \forall\overrightarrow{\beta^+}.\ M$. and by inversion there exists $\mu : (\overrightarrow{\beta^+} \cap \mathtt{fv}\ M) \leftrightarrow (\overrightarrow{\alpha^+} \cap \mathtt{fv}\ N)$ such that

- $\overrightarrow{\alpha^+} \cap \mathtt{fv}\ M = \emptyset$ and
- $N \simeq^D [\mu]M$

Let us assume that *vars* is disjoint from $\overrightarrow{\alpha^+}$ and $\overrightarrow{\beta^+}$ (we can always alpha-rename the bound variables). Then $\mathtt{ord}\ vars\ \mathtt{in} \forall\overrightarrow{\alpha^+}.\ N = \mathtt{ord}\ vars\ \mathtt{in}\ N$, $\mathtt{ord}\ vars\ \mathtt{in} \forall\overrightarrow{\alpha^+}.\ M = \mathtt{ord}\ vars\ \mathtt{in}\ M$ and by the induction hypothesis, $\mathtt{ord}\ vars\ \mathtt{in}\ N = \mathtt{ord}\ vars\ \mathtt{in}\ [\mu]M$. This way, it suffices tho show that $\mathtt{ord}\ vars\ \mathtt{in}\ [\mu]M = \mathtt{ord}\ vars\ \mathtt{in}\ M$. It holds by

Lemma 38 since *vars* is disjoint form the domain and the codomain of $\mu : (\overrightarrow{\beta^+} \cap \mathtt{fv}\ M) \leftrightarrow (\overrightarrow{\alpha^+} \cap \mathtt{fv}\ N)$ by assumption.

Case 5. The positive cases are proved symmetrically.

■

C.1.6 Normaliztaion

Observation 3 (Normalization is deterministic)**.** *If* $\mathtt{nf}\ (N) = M$ *and* $\mathtt{nf}\ (N) = M'$ *then* $M = M'$. *If* $\mathtt{nf}\ (P) = Q$ *and* $\mathtt{nf}\ (P) = Q'$ *then* $Q = Q'$. *This way, we can use normalization as a function.*

Proof By straightforward induction using Observation 2. ■

Lemma 40. *Free variables are not changed by the normalization*

- $\mathtt{fv}\ N = \mathtt{fv}\ \mathtt{nf}\ (N)$
- $\mathtt{fv}\ P = \mathtt{fv}\ \mathtt{nf}\ (P)$

Proof By mutual induction on N and P. The base cases ($(\textsc{Var}^{\textsc{nf}}_-)$ and $(\textsc{Var}^{\textsc{nf}}_+)$) are trivial; the congruent cases ($(\uparrow^{\textsc{nf}})$, $(\downarrow^{\textsc{nf}})$, and $(\to^{\textsc{nf}})$) are proved by the induction hypothesis.

Let us consider the case when the term is formed by $\forall$, that is the normalization judgment has a shape $\mathtt{nf}\ (\forall\overrightarrow{\alpha^+}.\ N) = \forall\overrightarrow{\alpha^{+\prime}}.\ N'$, where by inversion $\mathtt{nf}\ (N) = N'$ and $\mathtt{ord}\ \overrightarrow{\alpha^+}\ \mathtt{in}\ N' = \overrightarrow{\alpha^{+\prime}}$. By the induction hypothesis, $\mathtt{fv}\ N = \mathtt{fv}\ N'$. Since $\mathtt{fv}\ (\forall\overrightarrow{\alpha^+}.\ N) = \mathtt{fv}\ N \setminus \overrightarrow{\alpha^+}$, and $\mathtt{fv}\ (\forall\overrightarrow{\alpha^{+\prime}}.\ N') = \mathtt{fv}\ N' \setminus \overrightarrow{\alpha^{+\prime}}$, it is left to show that $\mathtt{fv}\ N \setminus \overrightarrow{\alpha^+} = \mathtt{fv}\ N \setminus \overrightarrow{\alpha^{+\prime}}$. By Lemma 39, $\overrightarrow{\alpha^{+\prime}} = \overrightarrow{\alpha^+} \cap \mathtt{fv}\ N' = \overrightarrow{\alpha^+} \cap \mathtt{fv}\ N$. Then $\mathtt{fv}\ N \setminus \overrightarrow{\alpha^+} = \mathtt{fv}\ N \setminus (\overrightarrow{\alpha^+} \cup \mathtt{fv}\ N)$ by set-theoretic properties, and thus, $\mathtt{fv}\ N \setminus \overrightarrow{\alpha^+} = \mathtt{fv}\ N \setminus \overrightarrow{\alpha^{+\prime}}$.

The case when the term is positive and formed by $\exists$ is symmetric. ■

Lemma 41 (Soundness of normalization)**.**

- $N \simeq^D \mathtt{nf}\ (N)$
- $P \simeq^D \mathtt{nf}\ (P)$

Proof Mutual induction on $\mathtt{nf}\ (N) = M$ and $\mathtt{nf}\ (P) = Q$. Let us consider how this judgment is formed:

Case 1. $(\textsc{Var}^{\textsc{nf}}_-)$ and $(\textsc{Var}^{\textsc{nf}}_+)$
By the corresponding equivalence rules.

Case 2. $(\uparrow^{\textsc{nf}})$, $(\downarrow^{\textsc{nf}})$, and $(\to^{\textsc{nf}})$
By the induction hypothesis and the corresponding congruent equivalence rules.

Case 3. $(\forall^{\textsc{nf}})$, i.e. $\mathtt{nf}\ (\forall\overrightarrow{\alpha^+}.\ N) = \forall\overrightarrow{\alpha^{+\prime}}.\ N'$
From the induction hypothesis, we know that $N \simeq^D N'$. In particular, by Lemma 26, $\mathtt{fv}\ N \equiv \mathtt{fv}\ N'$. Then by Lemma 35, $\overrightarrow{\alpha^{+\prime}} \equiv \overrightarrow{\alpha^+} \cap \mathtt{fv}\ N' \equiv \overrightarrow{\alpha^+} \cap \mathtt{fv}\ N$, and thus, $\overrightarrow{\alpha^{+\prime}} \cap \mathtt{fv}\ N' \equiv \overrightarrow{\alpha^+} \cap \mathtt{fv}\ N$.

To prove $\forall\overrightarrow{\alpha^+}.\, N \simeq^D \forall\overrightarrow{\alpha^+}.\, N'$, it suffices to provide a bijection $\mu : \overrightarrow{\alpha^+}' \cap \mathtt{fv}\, N' \leftrightarrow \overrightarrow{\alpha^+} \cap \mathtt{fv}\, N$ such that $N \simeq^D [\mu]N'$. Since these sets are equal, we take $\mu = id$.

Case 4. ($\exists^{\mathrm{NF}}$) Same as for case 3.

■

Corollary 14 (Normalization preserves well-formedness)**.**

+ $\Theta \vdash P \iff \Theta \vdash \mathtt{nf}\,(P)$,

− $\Theta \vdash N \iff \Theta \vdash \mathtt{nf}\,(N)$

Proof Immediately from Lemmas 28 and 41. ■

Corollary 15 (Normalization preserves well-formedness of substitution)**.**
$\Theta_2 \vdash \sigma : \Theta_1 \iff \Theta_2 \vdash \mathtt{nf}\,(\sigma) : \Theta_1$

Proof Let us prove the forward direction. Suppose that $\alpha^\pm \in \Theta_1$. Let us show that $\Theta_2 \vdash [\mathtt{nf}\,(\sigma)]\alpha^\pm$. By the definition of substitution normalization, $[\mathtt{nf}\,(\sigma)]\alpha^\pm = \mathtt{nf}\,([\sigma]\alpha^\pm)$. Then by Corollary 14, to show $\Theta_2 \vdash \mathtt{nf}\,([\sigma]\alpha^\pm)$, it suffices to show $\Theta_2 \vdash [\sigma]\alpha^\pm$, which holds by the assumption $\Theta_2 \vdash \sigma : \Theta_1$.

The backward direction is proved analogously. ■

Lemma 42 (Normalization preserves substitution signature)**.** *Suppose that σ is a substitution, Θ_1 and Θ_2 are contexts. Then $\Theta_2 \vdash \sigma : \Theta_1$ implies $\Theta_2 \vdash \mathtt{nf}\,(\sigma) : \Theta_1$.*

Proof Suppose that $\alpha^\pm \in \Theta_1$. Then by Corollary 14, $\Theta_2 \vdash \mathtt{nf}\,([\sigma]\alpha^\pm) = [\mathtt{nf}\,(\sigma)]\alpha^\pm$ is equivalent to $\Theta_2 \vdash [\sigma]\alpha^\pm$.

Suppose that $\alpha^\pm \notin \Theta_1$. $\Theta_2 \vdash \sigma : \Theta_1$ means that $[\sigma]\alpha^\pm = \alpha^\pm$, and then $[\mathtt{nf}\,(\sigma)]\alpha^\pm = \mathtt{nf}\,([\sigma]\alpha^\pm) = \mathtt{nf}\,(\alpha^\pm) = \alpha^\pm$. ■

Corollary 16 (Normalization is sound w.r.t. subtyping-induced equivalence)**.**

+ *if* $\Theta \vdash P$ *then* $\Theta \vdash P \simeq^{\leqslant} \mathtt{nf}\,(P)$,

− *if* $\Theta \vdash N$ *then* $\Theta \vdash N \simeq^{\leqslant} \mathtt{nf}\,(N)$.

Proof Immediately from Lemmas 29 and 41 and Corollary 14. ■

Corollary 17 (Normalization preserves subtyping)**.** *Assuming all the types are well-formed in context* Θ*,*

+ $\Theta \vdash P \geqslant Q \iff \Theta \vdash \mathtt{nf}\,(P) \geqslant \mathtt{nf}\,(Q)$,

− $\Theta \vdash N \leqslant M \iff \Theta \vdash \mathtt{nf}\,(N) \leqslant \mathtt{nf}\,(M)$.

Proof

+ $\Rightarrow$ Let us assume $\Theta \vdash P \geqslant Q$. By Corollary 16, $\Theta \vdash P \simeq^{\leqslant} \mathtt{nf}\,(P)$ and $\Theta \vdash Q \simeq^{\leqslant} \mathtt{nf}\,(Q)$, in particular, by inversion, $\Theta \vdash \mathtt{nf}\,(P) \geqslant P$ and $\Theta \vdash Q \geqslant \mathtt{nf}\,(Q)$. Then by transitivity of subtyping (Lemma 24), $\Theta \vdash \mathtt{nf}\,(P) \geqslant \mathtt{nf}\,(Q)$.

⇐ Let us assume $\Theta \vdash \mathtt{nf}\,(P) \geqslant \mathtt{nf}\,(Q)$. Also by Corollary 16 and inversion, $\Theta \vdash P \geqslant \mathtt{nf}\,(P)$ and $\Theta \vdash \mathtt{nf}\,(Q) \geqslant Q$. Then by the transitivity, $\Theta \vdash P \geqslant Q$.

– The negative case is proved symmetrically.

■

Corollary 18 (Normalization preserves ordering)**.** *For any vars,*

– $\mathtt{ord}\ \mathit{vars}\ \mathtt{in}\ \mathtt{nf}\,(N) = \mathtt{ord}\ \mathit{vars}\ \mathtt{in}\ M$

\+ $\mathtt{ord}\ \mathit{vars}\ \mathtt{in}\ \mathtt{nf}\,(P) = \mathtt{ord}\ \mathit{vars}\ \mathtt{in}\ Q$

Proof Immediately from Lemmas 39 and 41. ■

Lemma 43 (Distributivity of normalization over substitution)**.** *Normalization of a term distributes over substitution. Suppose that σ is a substitution, N and P are types. Then*

– $\mathtt{nf}\,([\sigma]N) = [\mathtt{nf}\,(\sigma)]\mathtt{nf}\,(N)$

\+ $\mathtt{nf}\,([\sigma]P) = [\mathtt{nf}\,(\sigma)]\mathtt{nf}\,(P)$

where $\mathtt{nf}\,(\sigma)$ *means pointwise normalization:* $[\mathtt{nf}\,(\sigma)]\alpha^- = \mathtt{nf}\,([\sigma]\alpha^-)$.

Proof Mutual induction on N and P.

Case 1. $N = \alpha^-$

$\mathtt{nf}\,([\sigma]N) = \mathtt{nf}\,([\sigma]\alpha^-) = [\mathtt{nf}\,(\sigma)]\alpha^-$.

$[\mathtt{nf}\,(\sigma)]\mathtt{nf}\,(N) = [\mathtt{nf}\,(\sigma)]\mathtt{nf}\,(\alpha^-) = [\mathtt{nf}\,(\sigma)]\alpha^-$.

Case 2. $P = \alpha^+$

Similar to case 1.

Case 3. If the type is formed by $\to$, $\uparrow$, or $\downarrow$, the required equality follows from the congruence of the normalization and substitution and the induction hypothesis. For example, if $N = P \to M$ then

$$
\begin{aligned}
\mathtt{nf}\,([\sigma]N) &= \mathtt{nf}\,([\sigma](P \to M)) \\
&= \mathtt{nf}\,([\sigma]P \to [\sigma]M) && \text{By congruence of substitution} \\
&= \mathtt{nf}\,([\sigma]P) \to \mathtt{nf}\,([\sigma]M) && \text{By congruence of normalization} \\
&= [\mathtt{nf}\,(\sigma)]\mathtt{nf}\,(P) \to [\mathtt{nf}\,(\sigma)]\mathtt{nf}\,(M) && \text{By induction hypothesis} \\
&= [\mathtt{nf}\,(\sigma)](\mathtt{nf}\,(P) \to \mathtt{nf}\,(M)) && \text{By congruence of substitution} \\
&= [\mathtt{nf}\,(\sigma)]\mathtt{nf}\,(P \to M) && \text{By congruence of normalization} \\
&= [\mathtt{nf}\,(\sigma)]\mathtt{nf}\,(N)
\end{aligned}
$$

Case 4. $N = \forall \overrightarrow{\alpha^+}.\, M$

$$
\begin{aligned}
[\mathtt{nf}\,(\sigma)]\mathtt{nf}\,(N) &= [\mathtt{nf}\,(\sigma)]\mathtt{nf}\,(\forall \overrightarrow{\alpha^+}.\, M) \\
&= [\mathtt{nf}\,(\sigma)]\forall \overrightarrow{\alpha^{+\prime}}.\, \mathtt{nf}\,(M) && \text{Where } \overrightarrow{\alpha^{+\prime}} = \mathtt{ord}\ \overrightarrow{\alpha^+}\ \mathtt{in}\ \mathtt{nf}\,(M) = \mathtt{ord}\ \overrightarrow{\alpha^+}\ \mathtt{in}\ M \\
& && \text{(the latter is by Corollary 18)}
\end{aligned}
$$

$$
\begin{aligned}
\mathtt{nf}\,([\sigma]N) &= \mathtt{nf}\,([\sigma]\forall\overrightarrow{\alpha^{+}}.\,M) \\
&= \mathtt{nf}\,(\forall\overrightarrow{\alpha^{+}}.\,[\sigma]M) && \text{Assuming } \overrightarrow{\alpha^{+}} \cap \Theta_1 = \emptyset \text{ and } \overrightarrow{\alpha^{+}} \cap \Theta_2 = \emptyset \\
&= \forall\overrightarrow{\beta^{+}}.\,\mathtt{nf}\,([\sigma]M) && \text{Where } \overrightarrow{\beta^{+}} = \mathtt{ord}\,\overrightarrow{\alpha^{+}}\,\mathtt{in}\,\mathtt{nf}\,([\sigma]M) = \mathtt{ord}\,\overrightarrow{\alpha^{+}}\,\mathtt{in}\,[\sigma]M \\
& && \text{(the latter is by Corollary 18)} \\
&= \forall\overrightarrow{\alpha^{+\prime}}.\,\mathtt{nf}\,([\sigma]M) && \text{By Lemma 38,} \\
& && \overrightarrow{\beta^{+}} = \overrightarrow{\alpha^{+\prime}} \text{ since } \overrightarrow{\alpha^{+}} \text{ is disjoint with } \Theta_1 \text{ and } \Theta_2 \\
&= \forall\overrightarrow{\alpha^{+\prime}}.\,[\mathtt{nf}\,(\sigma)]\mathtt{nf}\,(M) && \text{By the induction hypothesis}
\end{aligned}
$$

To show the alpha-equivalence of $[\mathtt{nf}\,(\sigma)]\forall\overrightarrow{\alpha^{+\prime}}.\,\mathtt{nf}\,(M)$ and $\forall\overrightarrow{\alpha^{+\prime}}.\,[\mathtt{nf}\,(\sigma)]\mathtt{nf}\,(M)$, we can assume that $\overrightarrow{\alpha^{+\prime}} \cap \Theta_1 = \emptyset$, and $\overrightarrow{\alpha^{+\prime}} \cap \Theta_2 = \emptyset$.

Case 5. $P = \exists\overrightarrow{\alpha^{-}}.\,Q$

Same as for case 4.

■

Corollary 19 (Commutativity of normalization and renaming)**.** *Normalization of a term commutes with renaming. Suppose that μ is a bijection between two sets of variables $\mu : A \leftrightarrow B$. Then*

- $\mathtt{nf}\,([\mu]N) = [\mu]\mathtt{nf}\,(N)$
- $\mathtt{nf}\,([\mu]P) = [\mu]\mathtt{nf}\,(P)$

Proof Immediately from Lemma 43, after noticing that $\mathtt{nf}\,(\mu) = \mu$. ■

Lemma 44 (Completeness of Normalization w.r.t. Declarative Equivalence)**.** *Normalization returns the same representative for equivalent types.*

- *If $N \simeq^{D} M$ then $\mathtt{nf}\,(N) = \mathtt{nf}\,(M)$,*
- *if $P \simeq^{D} Q$ then $\mathtt{nf}\,(P) = \mathtt{nf}\,(Q)$.*

Proof Mutual induction on $N \simeq^{D} M$ and $P \simeq^{D} Q$.

Case 1. $(\forall^{\simeq^{D}})$ From the normalization definition,

- $\mathtt{nf}\,(\forall\overrightarrow{\alpha^{+}}.\,N) = \forall\overrightarrow{\alpha^{+\prime}}.\,\mathtt{nf}\,(N)$ where $\overrightarrow{\alpha^{+\prime}}$ is $\mathtt{ord}\,\overrightarrow{\alpha^{+}}\,\mathtt{in}\,\mathtt{nf}\,(N)$
- $\mathtt{nf}\,(\forall\overrightarrow{\beta^{+}}.\,M) = \forall\overrightarrow{\beta^{+\prime}}.\,\mathtt{nf}\,(M)$ where $\overrightarrow{\beta^{+\prime}}$ is $\mathtt{ord}\,\overrightarrow{\beta^{+}}\,\mathtt{in}\,\mathtt{nf}\,(M)$

Let us take $\mu : (\overrightarrow{\beta^{+}} \cap \mathtt{fv}\,M) \leftrightarrow (\overrightarrow{\alpha^{+}} \cap \mathtt{fv}\,N)$ from the inversion of the equivalence judgment. Notice that from Lemmas 35 and 40, the domain and the codomain of μ can be written as $\mu : \overrightarrow{\beta^{+\prime}} \leftrightarrow \overrightarrow{\alpha^{+\prime}}$.

To show the alpha-equivalence of $\forall\overrightarrow{\alpha^{+\prime}}.\,\mathtt{nf}\,(N)$ and $\forall\overrightarrow{\beta^{+\prime}}.\,\mathtt{nf}\,(M)$, it suffices to prove that (i) $[\mu]\mathtt{nf}\,(M) = \mathtt{nf}\,(N)$ and
(ii) $[\mu]\overrightarrow{\beta^{+\prime}} = \overrightarrow{\alpha^{+\prime}}$.

(i) $[\mu]\mathtt{nf}\,(M) = \mathtt{nf}\,([\mu]M) = \mathtt{nf}\,(N)$. The first equality holds by Corollary 19, the second—by the induction hypothesis.

(ii)

$$
\begin{aligned}
[\mu]\overrightarrow{\beta^{+\prime}} &= [\mu]\mathtt{ord}\,\overrightarrow{\beta^{+}}\,\mathtt{in}\,\mathtt{nf}\,(M) && \text{by the definition of } \overrightarrow{\beta^{+\prime}}\\
&= [\mu]\mathtt{ord}\,(\overrightarrow{\beta^{+}} \cap \mathtt{fv}\,M)\,\mathtt{in}\,\mathtt{nf}\,(M) && \text{from Lemmas 36 and 40}\\
&= \mathtt{ord}\,[\mu](\overrightarrow{\beta^{+}} \cap \mathtt{fv}\,M)\,\mathtt{in}\,[\mu]\mathtt{nf}\,(M) && \text{by Lemma 37, because}\\
& && \overrightarrow{\alpha^{+}} \cap \mathtt{fv}\,N \cap \mathtt{fv}\,\mathtt{nf}\,(M) \subseteq \overrightarrow{\alpha^{+}} \cap \mathtt{fv}\,M = \emptyset\\
& && \overrightarrow{\alpha^{+}} \cap \mathtt{fv}\,N \cap (\overrightarrow{\beta^{+}} \cap \mathtt{fv}\,M) \subseteq \overrightarrow{\alpha^{+}} \cap \mathtt{fv}\,M = \emptyset\\
&= \mathtt{ord}\,[\mu](\overrightarrow{\beta^{+}} \cap \mathtt{fv}\,M)\,\mathtt{in}\,\mathtt{nf}\,(N) && \text{since } [\mu]\mathtt{nf}\,(M) = \mathtt{nf}\,(N) \text{ is proved}\\
&= \mathtt{ord}\,(\overrightarrow{\alpha^{+}} \cap \mathtt{fv}\,N)\,\mathtt{in}\,\mathtt{nf}\,(N) && \text{because } \mu \text{ is a bijection between}\\
& && \overrightarrow{\alpha^{+}} \cap \mathtt{fv}\,N \text{ and } \overrightarrow{\beta^{+}} \cap \mathtt{fv}\,M\\
&= \mathtt{ord}\,\overrightarrow{\alpha^{+}}\,\mathtt{in}\,\mathtt{nf}\,(N) && \text{from Lemmas 36 and 40}\\
&= \overrightarrow{\alpha^{+\prime}} && \text{by the definition of } \overrightarrow{\alpha^{+\prime}}
\end{aligned}
$$

Case 2. $(\exists^{\simeq^D})$ Same as for case 1.

Case 3. Other rules are congruent, and thus, proved by the corresponding congruent alpha-equivalence rule, which is applicable by the induction hypothesis.

■

Lemma 45 (Algorithmization of Declarative Equivalence)**.** *Declarative equivalence is the equality of normal forms.*

+ $P \simeq^D Q \iff \mathtt{nf}\,(P) = \mathtt{nf}\,(Q)$,
− $N \simeq^D M \iff \mathtt{nf}\,(N) = \mathtt{nf}\,(M)$.

Proof

+ Let us prove both directions separately.
  ⇒ exactly by Lemma 44,
  ⇐ from Lemma 41, we know $P \simeq^D \mathtt{nf}\,(P) = \mathtt{nf}\,(Q) \simeq^D Q$, then by transitivity (Lemma 27), $P \simeq^D Q$.
− For the negative case, the proof is the same.

■

Corollary 20 (Completeness of Normalization w.r.t. Subtyping-Induced Equivalence)**.** *Assuming all the types below are well-formed in* Θ*:*

+ *if* $\Theta \vdash P \simeq^{\leqslant} Q$ *then* $\mathtt{nf}\,(P) = \mathtt{nf}\,(Q)$,
− *if* $\Theta \vdash N \simeq^{\leqslant} M$ *then* $\mathtt{nf}\,(N) = \mathtt{nf}\,(M)$.

Proof Immediately from Lemmas 34 and 44. ∎

Lemma 46 (Idempotence of normalization)**.** *Normalization is idempotent*

− $\mathtt{nf}\,(\mathtt{nf}\,(N)) = \mathtt{nf}\,(N)$
+ $\mathtt{nf}\,(\mathtt{nf}\,(P)) = \mathtt{nf}\,(P)$

Proof By applying Lemma 44 to Lemma 41. ∎

Lemma 47. *The result of a substitution is normalized if and only if the initial type and the substitution are normalized.*

Suppose that σ *is a substitution* $\Theta_2 \vdash \sigma : \Theta_1$*,* P *is a positive type* ($\Theta_1 \vdash P$)*,* N *is a negative type* ($\Theta_1 \vdash N$)*. Then*

$$+\ [\sigma]P \text{ is normal} \iff \begin{cases} \sigma|_{\mathtt{fv}\,(P)} & \text{is normal} \\ P & \text{is normal} \end{cases}$$

$$-\ [\sigma]N \text{is normal} \iff \begin{cases} \sigma|_{\mathtt{fv}\,(N)} & \text{is normal} \\ N & \text{is normal} \end{cases}$$

Proof Mutual induction on $\Theta_1 \vdash P$ and $\Theta_1 \vdash N$.

Case 1. $N = \alpha^-$
Then N is always normal, and the normality of $\sigma|_{\alpha^-}$ by the definition means $[\sigma]\alpha^-$ is normal.

Case 2. $N = P \to M$

$$[\sigma](P \to M) \text{ is normal} \iff [\sigma]P \to [\sigma]M \text{ is normal} \qquad \text{by substitution congruence}$$

$$\iff \begin{cases} [\sigma]P & \text{is normal} \\ [\sigma]M & \text{is normal} \end{cases}$$

$$\iff \begin{cases} P & \text{is normal} \\ \sigma|_{\mathtt{fv}\,(P)} & \text{is normal} \\ M & \text{is normal} \\ \sigma|_{\mathtt{fv}\,(M)} & \text{is normal} \end{cases} \qquad \text{by the induction hypothesis}$$

$$\iff \begin{cases} P \to M & \text{is normal} \\ \sigma|_{\mathtt{fv}\,(P)\cup\mathtt{fv}\,(M)} & \text{is normal} \end{cases}$$

$$\iff \begin{cases} P \to M & \text{is normal} \\ \sigma|_{\mathtt{fv}\,(P\to M)} & \text{is normal} \end{cases}$$

Case 3. $N = \uparrow P$

By congruence and the inductive hypothesis, similar to case 2

Case 4. $N = \forall \overrightarrow{\alpha^{+}}.\ M$

$$
\begin{aligned}
& [\sigma](\forall \overrightarrow{\alpha^{+}}.\ M) \text{ is normal} \\
\iff & (\forall \overrightarrow{\alpha^{+}}.\ [\sigma]M) \text{ is normal} && \text{assuming } \overrightarrow{\alpha^{+}} \cap \Theta_1 = \emptyset \text{ and } \overrightarrow{\alpha^{+}} \cap \Theta_2 = \emptyset \\
\iff & \begin{cases} [\sigma]M \text{ is normal} \\ \mathtt{ord}\ \overrightarrow{\alpha^{+}}\ \mathtt{in}\ [\sigma]M = \overrightarrow{\alpha^{+}} \end{cases} && \text{by the definition of normalization} \\
\iff & \begin{cases} [\sigma]M \text{ is normal} \\ \mathtt{ord}\ \overrightarrow{\alpha^{+}}\ \mathtt{in}\ M = \overrightarrow{\alpha^{+}} \end{cases} && \text{by Lemma 38} \\
\iff & \begin{cases} \sigma|_{\mathtt{fv}\,(M)} \text{ is normal} \\ M \text{ is normal} \\ \mathtt{ord}\ \overrightarrow{\alpha^{+}}\ \mathtt{in}\ M = \overrightarrow{\alpha^{+}} \end{cases} && \text{by the induction hypothesis} \\
\iff & \begin{cases} \sigma|_{\mathtt{fv}\,(\forall \overrightarrow{\alpha^{+}}.\ M)} \text{ is normal} \\ \forall \overrightarrow{\alpha^{+}}.\ M \text{ is normal} \end{cases} && \text{since } \mathtt{fv}\,(\forall \overrightarrow{\alpha^{+}}.\ M) = \mathtt{fv}\,(M); \\
& && \text{by the definition of normalization}
\end{aligned}
$$

Case 5. $P = \ldots$

The positive cases are done in the same way as the negative ones.

■

Lemma 48 (Algorithmization of subtyping-induced equivalence)**.** *Mutual subtyping is the equality of normal forms. Assuming all the types below are well-formed in* Θ*:*

+ $\Theta \vdash P \simeq^{\leqslant} Q \iff \mathtt{nf}\,(P) = \mathtt{nf}\,(Q)$,
− $\Theta \vdash N \simeq^{\leqslant} M \iff \mathtt{nf}\,(N) = \mathtt{nf}\,(M)$.

Proof Let us prove the positive case, the negative case is symmetric. We prove both directions of $\iff$ separately:

$\Rightarrow$ exactly Corollary 20;

$\Leftarrow$ by Lemmas 29 and 45.

■

Corollary 21 (Substitution preserves declarative equivalence)**.** *Suppose that* σ *is a substitution. Then*

+ $P \simeq^{D} Q$ *implies* $[\sigma]P \simeq^{D} [\sigma]Q$
− $N \simeq^{D} M$ *implies* $[\sigma]N \simeq^{D} [\sigma]M$

Proof

$$
\begin{aligned}
P \simeq^{D} Q &\Rightarrow \mathtt{nf}\,(P) = \mathtt{nf}\,(Q) && \text{by Lemma 48} \\
&\Rightarrow [\mathtt{nf}\,(\sigma)]\mathtt{nf}\,(P) = [\mathtt{nf}\,(\sigma)]\mathtt{nf}\,(Q)
\end{aligned}
$$

$\Rightarrow \mathtt{nf}([\sigma]P) = \mathtt{nf}([\sigma]Q)$ by Lemma 43

$\Rightarrow [\sigma]P \simeq^{D} [\sigma]Q$ by Lemma 48

■

C.2 Relation to System F

Lemma 51 (Subtyping elaboration term can be removed)**.**

- *For any* Θ, N, *and* M, $\Theta \vdash N \leqslant M$ *holds if and only if there exists* t *such that* $\Theta \vdash N \leqslant M \rightsquigarrow t$;
- + *For any* Θ, P, *and* Q, $\Theta \vdash P \geqslant Q$ *holds if and only if there exists* t *such that* $\Theta \vdash P \geqslant Q \rightsquigarrow t$.

Proof We prove it separately in both directions of the implication. both of the implications are proved by simple induction on the judgment. ■

Observation 4 (Type depolarization distributes over substitution)**.**

- + $|[\sigma]N| = [|\sigma|]|N|$,
- − $|[\sigma]P| = [|\sigma|]|P|$.

Proof By mutual induction on the type N and P. ■

Lemma 52 (Soundness of Subtyping Elaboration)**.**

- − *If* $\Theta \vdash N \leqslant M \rightsquigarrow t$ *then* $|\Theta|; \cdot \vdash t : |N| \to |M|$;
- + *if* $\Theta \vdash P \geqslant Q \rightsquigarrow t$ *then* $|\Theta|; \cdot \vdash t : |Q| \to |P|$.

Proof We prove it by (mutual) induction on $\Theta \vdash N \leqslant M \rightsquigarrow t$ and $\Theta \vdash P \geqslant Q \rightsquigarrow t$. Let us consider the last rule applied to infer this judgment.

Case 1. $(\textsc{Var}^{\rightsquigarrow}_{\leqslant})$ Notice that $|\alpha^{-}| = \alpha$. Then $|\Theta|; \cdot \vdash \lambda x.\, x : \alpha \to \alpha$ immediately by (λ^{F}) and $(\textsc{Var}^{\mathsf{F}})$.

Case 2. $(\textsc{Var}^{\rightsquigarrow}_{\geqslant})$ This case is symmetric to the previous one.

Case 3. $(\uparrow^{\rightsquigarrow}_{\leqslant})$ Notice that $|\uparrow P| = |P|$, and by induction hypothesis $\Theta \vdash Q \geqslant P \rightsquigarrow t$ implies $|\Theta|; \cdot \vdash t : |P| \to |Q|$.

Case 4. $(\downarrow^{\rightsquigarrow}_{\geqslant})$ This case is symmetric to the previous one.

Case 5. $(\to^{\rightsquigarrow}_{\leqslant})$ We need to show that $|\Theta|; \cdot \vdash \lambda x.\, \lambda y.\, t'\ (x\ (t\ y)) : (|P| \to |N|) \to |Q| \to |M|$. By induction hypothesis applied to the premises, we know that $|\Theta|; \cdot \vdash t : |Q| \to |P|$ and $|\Theta|; \cdot \vdash t' : |N| \to |M|$. Then the required typing judgment follows from standard rules of System F.

Case 6. $(\forall^{\rightsquigarrow}_{\leqslant})$ We need to show that $|\Theta|; \cdot \vdash \lambda x.\, \Lambda \vec{\beta}.\, t\ x : (\forall \vec{\alpha}.\, |N|) \to \forall \vec{\beta}.\, |M|$. By the induction hypothesis applied to the premise, we know that $|\Theta|; \cdot \vdash t : |[\sigma]N| \to |M|$,

that we rewrite as $|\Theta|; \cdot \vdash t \colon [|\sigma|]|N| \to |M|$ by Observation 4. Then the required typing judgment follows from the standard rules of System F notice that $|\Theta|, \vec{\beta}; x : \forall \vec{\alpha}.\, |N| \vdash x \colon \forall \vec{\alpha}.\, |N|$ implies $|\Theta|, \vec{\beta}; x : \forall \vec{\alpha}.\, |N| \vdash x \colon [|\sigma|]|N|$ by ($\mathsf{TApp}^{\mathsf{F}}$).

Case 7. ($\exists^{\rightsquigarrow}_{\geqslant}$) We need to show that $|\Theta|; \cdot \vdash \lambda x.\, \mathsf{unpack}\ (\vec{\beta}, y) = x\ ;\ \mathsf{pack}\ (t\ y)\ \mathsf{as}\ \exists \vec{\alpha}.\, |P| \colon |\exists \overrightarrow{\beta^-}.\, Q| \to |\exists \overrightarrow{\alpha^-}.\, P|$. By definition, $|\exists \overrightarrow{\beta^-}.\, Q| \to |\exists \overrightarrow{\alpha^-}.\, P| = \exists \vec{\beta}.\, |Q| \to \exists \vec{\alpha}.\, |P|$. Then by applying ($\lambda^{\mathsf{F}}$) and the admissible rules ($\mathsf{Unpack}^{\mathsf{F}}$) and ($\mathsf{Pack}^{\mathsf{F}}$), it suffices to show $|\Theta|, \vec{\beta}; y : |Q| \vdash (t\ y) \colon [|\sigma|]|P|$ and that $|\Theta|, \vec{\beta} \vdash |\sigma| \colon \vec{\alpha}$. The latter follows from $\Theta, \overrightarrow{\beta^-} \vdash \sigma : \overrightarrow{\alpha^-}$. The former holds by ($\mathsf{App}^{\mathsf{F}}$) and the induction hypothesis applied to the premise $(|\Theta|; \cdot \vdash t \colon |Q| \to |[\sigma]P|$, and by Observation 4, $|[\sigma]P| = [|\sigma|]|P|)$.

■

Lemma 53 (Soundness of $\mathrm{F}^{\pm}_{\exists}$ w.r.t. System F)**.** *A judgment inferred by $\mathrm{F}^{\pm}_{\exists}$ is derivable in System F.*

- \+ *If* $\Theta; \Gamma \vdash v \colon P \rightsquigarrow t$ *then* $|\Theta|; |\Gamma| \vdash t \colon |P|$*;*
- − *if* $\Theta; \Gamma \vdash c \colon N \rightsquigarrow t$ *then* $|\Theta|; |\Gamma| \vdash t \colon |N|$*;*
- • *if* $\Theta; \Gamma \vdash N \bullet \vec{v} \Rrightarrow M \rightsquigarrow e; \vec{t}$ *then* $|\Theta|; |\Gamma|, x : |N| \vdash e\ (x\, \vec{t}) \colon |M|$.

Proof We prove it by (mutual) induction on the derivation of $\Theta; \Gamma \vdash v \colon P \rightsquigarrow t$ and $\Theta; \Gamma \vdash c \colon N \rightsquigarrow t$.

Case 1. ($\to^{\rightsquigarrow}_{\bullet\Rrightarrow}$) Suppose that $\Theta; \Gamma \vdash Q \to N \bullet v, \vec{v} \Rrightarrow M \rightsquigarrow e'; e\ t, \vec{t}$. Then we know

- $\Theta; \Gamma \vdash v \colon P \rightsquigarrow t$, and then by the induction hypothesis, $|\Theta|; |\Gamma| \vdash t \colon |P|$;
- $\Theta \vdash Q \geqslant P \rightsquigarrow e$, and then by the soundness of subtyping elaboration (Lemma 52), $|\Theta|; \cdot \vdash e \colon |P| \to |Q|$;
- $\Theta; \Gamma \vdash N \bullet \vec{v} \Rrightarrow M \rightsquigarrow e'; \vec{t}$, and then by the induction hypothesis, $|\Theta|; |\Gamma|, x : |N| \vdash e'\ (x\, \vec{t}) \colon |M|$.

We wish to show that $|\Theta|; |\Gamma|, f : |Q| \to |N| \vdash e'\ (f(e\ t, \vec{t})) \colon |M|$, that is $|\Theta|; |\Gamma|, f : |Q| \to |N| \vdash e'\ (f\ (e\ t)\, \vec{t}) \colon |M|$. By substitution applied to case 1, it suffices to show that $|\Theta|; |\Gamma|, f : |Q| \to |N| \vdash f\ (e\ t) \colon |M|$, which is inferred by multiple applications of ($\mathsf{App}^{\mathsf{F}}$) to the judgments stated above.

Case 2. ($\forall^{\rightsquigarrow}_{\bullet\Rrightarrow}$) Then we know that $\Theta; \Gamma \vdash \forall \overrightarrow{\alpha^+}.\, N \bullet \vec{v} \Rrightarrow M \rightsquigarrow e; \vec{t}$, and by inversion: $\Theta; \Gamma \vdash [\sigma]N \bullet \vec{v} \Rrightarrow M \rightsquigarrow e; \vec{t}$, where σ is a substitution from $\overrightarrow{\alpha^+}$ to Θ. By the induction hypothesis, we know that $|\Theta|; |\Gamma|, x : [|\sigma|]|N| \vdash e\ (x\, \vec{t}) \colon |M|$.

We wish to show that $|\Theta|; |\Gamma|, x : \forall \vec{\alpha}.\, |N| \vdash e\ (x\, \vec{t}) \colon |M|$. To do that, we can modify the inference tree of $|\Theta|; |\Gamma|, x : [|\sigma|]|N| \vdash e\ (x\, \vec{t}) \colon |M|$ to infer $|\Theta|; |\Gamma|, x : \forall \vec{\alpha}.\, |N| \vdash e\ (x\, \vec{t}) \colon |M|$. The only thing we need to change is every time the inference asks for the type of x in the context (i.e., in the leaf rule ($\textsc{Var}^{\mathsf{F}}$)), we use the combination of ($\mathsf{TApp}^{\mathsf{F}}$) and ($\textsc{Var}^{\mathsf{F}}$) instead of only ($\textsc{Var}^{\mathsf{F}}$) to infer $|\Theta|; |\Gamma|, x : \forall \vec{\alpha}.\, |N| \vdash x \colon [|\sigma|]|N|$.

Case 3. $(\emptyset^{\rightsquigarrow}_{\bullet\Rightarrow})$ Then we know $\Theta; \Gamma \vdash N \bullet \Rightarrow N' \rightsquigarrow e; \cdot$. By inversion, we have $\Theta \vdash N' \leqslant N \rightsquigarrow e$, which by the soundness of subtyping elaboration (Lemma 52), means $|\Theta|; \cdot \vdash e : |N| \to |N'|$.

Then the desired $|\Theta|; |\Gamma|, x : |N| \vdash e\, x : |N'|$ follows from the standard rules of System F.

Case 4. $(\textsc{Var}^{\rightsquigarrow})$, which means we wish to prove that $\Theta; \Gamma \vdash x : P \rightsquigarrow x$ implies $|\Theta|; |\Gamma| \vdash x : |P|$. By inversion, $x : P \in \Gamma$, and thus, by definition of context depolarization, $x : |P| \in |\Gamma|$. Then by $(\textsc{Var}^{\mathsf{F}})$, we infer $|\Theta|; |\Gamma| \vdash x : |P|$.

Case 5. $(\{\}^{\rightsquigarrow})$ Then we know that $\Theta; \Gamma \vdash \{c\} : \downarrow N \rightsquigarrow t$ and by the inversion, $\Theta; \Gamma \vdash c : N \rightsquigarrow t$. Then the desired $|\Theta|; |\Gamma| \vdash t : |N|$ follows from the induction hypothesis.

Case 6. $(\textsc{ret}^{\rightsquigarrow})$ The proof is symmetric to the previous case.

Case 7. $(\textsc{ann}^{\rightsquigarrow}_{+})$ Then we know that $\Theta; \Gamma \vdash (v : Q) : Q \rightsquigarrow e\ t$ and by inversion:

- $\Theta; \Gamma \vdash v : P \rightsquigarrow t$, and then by the induction hypothesis, $|\Theta|; |\Gamma| \vdash t : |P|$;
- $\Theta \vdash Q \geqslant P \rightsquigarrow e$, and then by the soundness of subtyping elaboration (Lemma 52), $|\Theta|; \cdot \vdash e : |P| \to |Q|$.

The desired $|\Theta|; |\Gamma| \vdash e\, t : |Q|$ follows by $(\textsc{App}^{\mathsf{F}})$ applied to the judgement stated above.

Case 8. For $(\textsc{ann}^{\rightsquigarrow}_{-})$, $(\simeq^{\rightsquigarrow}_{+})$ and $(\simeq^{\rightsquigarrow}_{-})$ the proof is analogous to the previous case.

Case 9. $(\lambda^{\rightsquigarrow})$ We know that $\Theta; \Gamma \vdash \lambda x : P.\, c : P \to N \rightsquigarrow \lambda x.\, t$ and by inversion: $\Theta; \Gamma, x : P \vdash c : N \rightsquigarrow t$, then by the induction hypothesis, $|\Theta|; |\Gamma|, x : |P| \vdash t : |N|$. By applying (λ^{F}) to this judgment, we infer the desired $|\Theta|; |\Gamma| \vdash \lambda x.\, t : |P| \to |N|$.

Case 10. $(\Lambda^{\rightsquigarrow})$ We know that $\Theta; \Gamma \vdash \Lambda\alpha^{+}.\, c : \forall\alpha^{+}.\, N \rightsquigarrow \Lambda\alpha.\, t$ and by inversion: $\Theta, \alpha^{+}; \Gamma \vdash c : N \rightsquigarrow t$, then by the induction hypothesis, $|\Theta|, \alpha; |\Gamma| \vdash t : |N|$. By applying (Λ^{F}) to this judgment, we infer $|\Theta|; |\Gamma| \vdash \Lambda\alpha.\, t : \forall\alpha.\, |N|$, that is $|\Theta|; |\Gamma| \vdash \Lambda\alpha.\, t : |\forall\alpha^{+}.\, N|$.

Case 11. $(\textsc{let}^{\rightsquigarrow}_{\mathrm{c}})$ We know that $\Theta; \Gamma \vdash \mathtt{let}\, x : P = c;\ c' : N \rightsquigarrow \mathtt{let}\, x = (e\ t)\, ;\, t'$ and by inversion:

- $\Theta; \Gamma \vdash c : M \rightsquigarrow t$, and then by the induction hypothesis, $|\Theta|; |\Gamma| \vdash t : |M|$;
- $\Theta \vdash M \leqslant \uparrow P \rightsquigarrow e$, and then by the soundness of subtyping elaboration (Lemma 52), $|\Theta|; \cdot \vdash e : |M| \to |P|$;
- $\Theta; \Gamma, x : P \vdash c' : N \rightsquigarrow t'$, and then by the induction hypothesis, $|\Theta|; |\Gamma|, x : |P| \vdash t' : |N|$.

To infer $|\Theta|; |\Gamma| \vdash \mathtt{let}\, x = (e\ t)\, ;\, t' : |N|$, we apply the admissible System F rule $(\textsc{Let}^{\mathsf{F}})$. Both premises hold: $|\Theta|; |\Gamma| \vdash e\, t : |P|$ follows form $(\textsc{App}^{\mathsf{F}})$, and $|\Theta|; |\Gamma|, x : |P| \vdash t' : |N|$ holds as stated above.

Case 12. $(\textsc{let}^{\rightsquigarrow})$ We know that $\Theta; \Gamma \vdash \mathtt{let}\, x = v;\ c : N \rightsquigarrow \mathtt{let}\, x = t\, ;\, t'$ and by inversion:

- $\Theta; \Gamma \vdash v : P \rightsquigarrow t$ implying $|\Theta|; |\Gamma| \vdash t : |P|$; and
- $\Theta; \Gamma, x : P \vdash c : N \rightsquigarrow t'$ implying $|\Theta|; |\Gamma|, x : |P| \vdash t' : |N|$.

Then the desired $|\Theta|; |\Gamma| \vdash \mathtt{let}\, x = t\, ; t' : |N|$ follows by the admissible System F rule ($\mathsf{Let}^{\mathsf{F}}$) applied to the judgments above.

Case 13. ($\textsc{let}^{\rightsquigarrow}_{@}$) We know that $\Theta; \Gamma \vdash \mathtt{let}\, x = v(\vec{v})\, ; c : N \rightsquigarrow \mathtt{let}\, x = (e\ (t'\ \vec{t}))\, ; t$ and by inversion, in particular, we have:

- $\Theta; \Gamma \vdash v : \downarrow M \rightsquigarrow t'$, and then by the induction hypothesis, $|\Theta|; |\Gamma| \vdash t' : |M|$;
- $\Theta; \Gamma \vdash M \bullet \vec{v} \Rrightarrow \uparrow Q \rightsquigarrow e; \vec{t}$, and then by the induction hypothesis, $|\Theta|; |\Gamma|, x : |M| \vdash e\ (x\ \vec{t}) : |Q|$;
- $\Theta; \Gamma, x : Q \vdash c : N \rightsquigarrow t$, and then by the induction hypothesis, $|\Theta|; |\Gamma|, x : |Q| \vdash t : |N|$.

We wish to show that $|\Theta|; |\Gamma| \vdash \mathtt{let}\, x = (e\ (t'\ \vec{t}))\, ; t : |N|$. By applying ($\mathsf{Let}^{\mathsf{F}}$), we reduce this to $|\Theta|; |\Gamma| \vdash e\ (t'\ \vec{t}) : |Q|$, which holds by substitution t' for x in $|\Theta|; |\Gamma|, x : |M| \vdash e\ (x\ \vec{t}) : |Q|$ (which is possible since $|\Theta|; |\Gamma| \vdash t' : |M|$).

Case 14. ($\textsc{let}^{\rightsquigarrow}_{:@}$) By inversion of $\Theta; \Gamma \vdash \mathtt{let}\, x : P = v(\vec{v})\, ; c : N \rightsquigarrow \mathtt{let}\, x = e'\ (e\ (t'\ \vec{t}))\, ; t$, we know that

- $\Theta; \Gamma \vdash v : \downarrow M \rightsquigarrow t'$, and then by the induction hypothesis, $|\Theta|; |\Gamma| \vdash t' : |M|$;
- $\Theta; \Gamma \vdash M \bullet \vec{v} \Rrightarrow M' \rightsquigarrow e; \vec{t}$, and then by the induction hypothesis, $|\Theta|; |\Gamma|, x : |M| \vdash e\ (x\ \vec{t}) : |M'|$. By substitution, it implies $|\Theta|; |\Gamma| \vdash e\ (t'\ \vec{t}) : |M'|$.
- $\Theta \vdash M' \leqslant \uparrow P \rightsquigarrow e'$, and then by the soundness of subtyping elaboration (Lemma 52), $|\Theta|; \cdot \vdash e' : |M'| \to |P|$;
- $\Theta; \Gamma, x : P \vdash c : N \rightsquigarrow t$, and then by the induction hypothesis, $|\Theta|; |\Gamma|, x : |P| \vdash t : |N|$.

To infer $|\Theta|; |\Gamma| \vdash \mathtt{let}\, x = e'\ (e\ (t'\ \vec{t}))\, ; t : |N|$, we apply ($\mathsf{Let}^{\mathsf{F}}$), so it is left to show that $|\Theta|; |\Gamma| \vdash e'\ (e\ (t'\ \vec{t})) : |P|$, which follows from ($\mathsf{App}^{\mathsf{F}}$) and the judgments stated above.

Case 15. ($\textsc{let}^{\rightsquigarrow}_{\exists}$) By inversion of $\Theta; \Gamma \vdash \mathtt{let}^{\exists}(\vec{\alpha}, x) = v\, ; c : N \rightsquigarrow \mathtt{unpack}\ (\vec{\alpha}, x) = t\, ; t'$, we have:

- $\Theta; \Gamma \vdash v : \exists \vec{\alpha}.\, P \rightsquigarrow t$, and then by the induction hypothesis, $|\Theta|; |\Gamma| \vdash t : \exists \vec{\alpha}.\, |P|$;
- $\Theta; \Gamma, x : P \vdash c : N \rightsquigarrow t'$, and then by the induction hypothesis, $|\Theta|; |\Gamma|, x : |P| \vdash t' : |N|$;
- $\Theta \vdash N$ implying $|\Theta| \vdash |N|$.

To infer the desired $|\Theta|; |\Gamma| \vdash \mathtt{unpack}\ (\vec{\alpha}, x) = t\, ; t' : |N|$, we apply ($\mathsf{Unpack}^{\mathsf{F}}$): all its premises hold, as noted above.

■

Lemma 54 (Polarization commutes with substitution). $\downarrow[A/\alpha]T\downarrow = [\downarrow A\downarrow/\alpha^+]\downarrow T\downarrow$

Proof We prove it by induction on T. Each congruent case is proven by congruent rewriting, and applying the induction hypothesis.

- **Case 1**. $T = \alpha$. Then $\downarrow[A/\alpha]T\downarrow = \downarrow[A/\alpha]\alpha\downarrow = \downarrow A\downarrow$ and $[\downarrow A\downarrow/\alpha^+]\downarrow T\downarrow = [\downarrow A\downarrow/\alpha^+]\alpha^+ = \downarrow A\downarrow$.
- **Case 2**. $T = \alpha_0 \neq \alpha$. Then $\downarrow[A/\alpha]T\downarrow = \downarrow[A/\alpha]\alpha_0\downarrow = \downarrow\alpha_0\downarrow = \alpha_0^+$ and $[\downarrow A\downarrow/\alpha^+]\downarrow T\downarrow = [\downarrow A\downarrow/\alpha^+]\alpha_0^+ = \downarrow\alpha_0\downarrow$.
- **Case 3**. $T = B_1 \to B_1$. By the induction hypothesis, $\downarrow[A/\alpha]B_i\downarrow = [\downarrow A\downarrow/\alpha^+]\downarrow B_i\downarrow$ for $i = 1, 2$. Then $\downarrow[A/\alpha]T\downarrow = \downarrow[A/\alpha](B_1 \to B_2)\downarrow = \downarrow[A/\alpha]B_1 \to [A/\alpha]B_2\downarrow = \downarrow(\downarrow[A/\alpha]B_1\downarrow \to \uparrow\downarrow[A/\alpha]B_2\downarrow) = \downarrow([\downarrow A\downarrow/\alpha^+]\downarrow B_1\downarrow \to \uparrow[\downarrow A\downarrow/\alpha^+]\downarrow B_2\downarrow) = [\downarrow A\downarrow/\alpha^+]\downarrow(\downarrow B_1\downarrow \to \uparrow\downarrow B_2\downarrow) = [\downarrow A\downarrow/\alpha^+]\downarrow B_1 \to B_2\downarrow = [\downarrow A\downarrow/\alpha^+]\downarrow T\downarrow$.
- **Case 4**. $T = \forall\alpha_0.\, T_0$. By the induction hypothesis, $\downarrow[A/\alpha]T_0\downarrow = [\downarrow A\downarrow/\alpha^+]\downarrow T_0\downarrow$. Then $\downarrow[A/\alpha]T\downarrow = \downarrow[A/\alpha]\forall\alpha_0.\, T_0\downarrow = \downarrow\forall\alpha_0.\, [A/\alpha]T_0\downarrow = \downarrow\forall\alpha_0^+.\uparrow\downarrow[A/\alpha]T_0\downarrow = \downarrow\forall\alpha_0^+.\uparrow[\downarrow A\downarrow/\alpha^+]\downarrow T_0\downarrow = [\downarrow A\downarrow/\alpha^+]\downarrow\forall\alpha_0^+.\uparrow\downarrow T_0\downarrow = [\downarrow A\downarrow/\alpha^+]\downarrow\forall\alpha_0.\, T_0\downarrow = [\downarrow A\downarrow/\alpha^+]\downarrow T\downarrow$.

■

Observation 5. *For any* Θ, Γ, t, *and* T, *there exists* c *such that* $\Theta;\Gamma \vdash t : T \rightsquigarrow^{\pm} c$ *if and only if* $\Theta;\Gamma \vdash t : T$.

Proof We prove it separately in both directions of the implication.

The erasure of the computation is proved by induction on the derivation of $\Theta;\Gamma \vdash t : T \rightsquigarrow^{\pm} c$. Notice that the inference rules of Fig. 26 are obtained by removing the resulting computation from the judgement of the rules of Fig. 31.

The other direction is proved by straightforward induction on the typing derivation $\Theta;\Gamma \vdash t : T$. ■

Lemma 55 (Type polarization agrees with well-formedness). *If* $\Theta \vdash T$ *then* $\downarrow\Theta\downarrow \vdash \downarrow T\downarrow$.

Proof We prove it by induction on $\Theta \vdash T$. Let us consider the last rule applied to infer this judgment, or, equivalently, the shape of $\Theta \vdash T$.

- **Case 1**. $\Theta \vdash \alpha$. By inversion, $\alpha \in \Theta$, which implies $\downarrow\alpha\downarrow = \alpha^+ \in \downarrow\Theta\downarrow$. Then $\downarrow\Theta\downarrow \vdash \downarrow\alpha\downarrow$ by $(\textsc{Var}_+^{\textsc{WF}})$;
- **Case 2**. $\Theta \vdash A \to B$. By inversion, $\Theta \vdash A$ and $\Theta \vdash B$, which by the induction hypothesis implies $\downarrow\Theta\downarrow \vdash \downarrow A\downarrow$ and $\downarrow\Theta\downarrow \vdash \downarrow B\downarrow$. Then $\downarrow\Theta\downarrow \vdash \downarrow(\downarrow A\downarrow \to \uparrow\downarrow B\downarrow) = \downarrow A \to B\downarrow$ by $(\downarrow^{\textsc{WF}})$, $(\uparrow^{\textsc{WF}})$, and $(\to^{\textsc{WF}})$;
- **Case 3**. $\Theta \vdash \forall\vec{\alpha}.\, T$. By inversion, $\Theta, \alpha \vdash T$, which by the induction hypothesis implies $\downarrow\Theta\downarrow, \alpha^+ \vdash \downarrow T\downarrow$. Then $\downarrow\Theta\downarrow \vdash \forall\alpha^+.\uparrow\downarrow T\downarrow = \downarrow\forall\alpha.\, T\downarrow$ by $(\forall^{\textsc{WF}})$, and $(\uparrow^{\textsc{WF}})$.

■

Lemma 56 (Polarization preserves typing). *If* $\Theta;\Gamma \vdash t : T \rightsquigarrow^{\pm} c$ *then* $\downarrow\Theta\downarrow;\downarrow\Gamma\downarrow \vdash c : \uparrow\downarrow T\downarrow$.

Proof We prove it by induction on $\Theta; \Gamma \vdash t : T \rightsquigarrow^{\pm} c$. Let us consider the last rule applied to infer this judgment.

Case 1. ($\text{VAR}^{\rightsquigarrow^{\pm}}$), in which case we wish to prove that if $\Theta; \Gamma \vdash x : T \rightsquigarrow^{\pm} \texttt{return}\, x$ then $\lfloor\Theta\rfloor; \lfloor\Gamma\rfloor \vdash \texttt{return}\, x : \uparrow\lfloor T\rfloor$.

By inversion of the given judgement, $x : T \in \Gamma$, and thus, $x : \lfloor T\rfloor \in \lfloor\Gamma\rfloor$. We apply ($\text{VAR}^{\text{INF}}$) to infer $\lfloor\Theta\rfloor; \lfloor\Gamma\rfloor \vdash x : \lfloor T\rfloor$, and then ($\text{RET}^{\text{INF}}$) to infer $\lfloor\Theta\rfloor; \lfloor\Gamma\rfloor \vdash \texttt{return}\, x : \uparrow\lfloor T\rfloor$.

Case 2. ($\lambda^{\rightsquigarrow^{\pm}}$), in which case we wish to prove that if $\Theta; \Gamma \vdash \lambda x.\, t : A \to B \rightsquigarrow^{\pm} \texttt{return}\, \{\lambda x : \lfloor A\rfloor .\, c\}$ then $\lfloor\Theta\rfloor; \lfloor\Gamma\rfloor \vdash \texttt{return}\, \{\lambda x : \lfloor A\rfloor .\, c\} : \uparrow\lfloor A \to B\rfloor$.

The premise of the given judgment is $\Theta; \Gamma, x : A \vdash t : B \rightsquigarrow^{\pm} c$, which by the induction hypothesis implies $\lfloor\Theta\rfloor; \lfloor\Gamma, x : A\rfloor \vdash c : \uparrow\lfloor B\rfloor$, which we rewrite as $\lfloor\Theta\rfloor; \lfloor\Gamma\rfloor, x : \lfloor A\rfloor \vdash c : \uparrow\lfloor B\rfloor$. Also, the soundness of System F typing (together with Observation 5) imply that the inferred type $A \to B$ is well-formed in Θ, which means that $\lfloor A\rfloor$ is well-formed in $\lfloor\Theta\rfloor$ by **??**. Then by (λ^{INF}) we infer $\lfloor\Theta\rfloor; \lfloor\Gamma\rfloor \vdash \lambda x : \lfloor A\rfloor .\, c : \lfloor A\rfloor \to \uparrow\lfloor B\rfloor$, and then by ($\{\}^{\text{INF}}$) and ($\text{RET}^{\text{INF}}$), $\lfloor\Theta\rfloor; \lfloor\Gamma\rfloor \vdash \texttt{return}\, \{\lambda x : \lfloor A\rfloor .\, c\} : \uparrow\downarrow(\lfloor A\rfloor \to \uparrow\lfloor B\rfloor)$, which can be rewritten to the required judgment since $\downarrow(\lfloor A\rfloor \to \uparrow\lfloor B\rfloor) = \lfloor A \to B\rfloor$ by definition.

Case 3. ($\Lambda^{\rightsquigarrow^{\pm}}$). The case is similar to the previous one. We wish to prove that if $\Theta; \Gamma \vdash \Lambda\alpha.\, t : \forall\alpha.\, T \rightsquigarrow^{\pm} \texttt{return}\, \{\Lambda\alpha^{+} .\, c\}$ then $\lfloor\Theta\rfloor; \lfloor\Gamma\rfloor \vdash \texttt{return}\, \{\Lambda\alpha^{+} .\, c\} : \uparrow\lfloor\forall\alpha.\, T\rfloor$.

By inversion of the given judgment, we have: $\Theta, \alpha; \Gamma \vdash t : T \rightsquigarrow^{\pm} c$, and by the induction hypothesis, $\lfloor\Theta\rfloor, \alpha^{+}; \lfloor\Gamma\rfloor \vdash c : \uparrow\lfloor T\rfloor$,

Then we apply (Λ^{INF}) to infer $\lfloor\Theta\rfloor; \lfloor\Gamma\rfloor \vdash \Lambda\alpha^{+} .\, c : \forall\alpha^{+} .\, \uparrow\lfloor T\rfloor$, and then ($\{\}^{\text{INF}}$) with ($\text{RET}^{\text{INF}}$) to infer $\lfloor\Theta\rfloor; \lfloor\Gamma\rfloor \vdash \texttt{return}\, \{\Lambda\alpha^{+} .\, c\} : \uparrow\downarrow\forall\alpha^{+} .\, \uparrow\lfloor T\rfloor$. Finally, notice that $\downarrow\forall\alpha^{+} .\, \uparrow\lfloor T\rfloor = \lfloor\forall\alpha.\, T\rfloor$ by definition.

Case 4. ($\text{App}^{\rightsquigarrow^{\pm}}$). By inversion of the given judgment, we have:

- $\Theta; \Gamma \vdash t : A \to B \rightsquigarrow^{\pm} c$, and by the induction hypothesis, $\lfloor\Theta\rfloor; \lfloor\Gamma\rfloor \vdash c : \uparrow\lfloor A \to B\rfloor$, which is equivalent to $\lfloor\Theta\rfloor; \lfloor\Gamma\rfloor \vdash c : \uparrow\downarrow(\lfloor A\rfloor \to \uparrow\lfloor B\rfloor)$.
- $\Theta; \Gamma \vdash t' : A \rightsquigarrow^{\pm} c'$, and by the induction hypothesis, $\lfloor\Theta\rfloor; \lfloor\Gamma\rfloor \vdash c' : \uparrow\lfloor A\rfloor$.

To infer $\lfloor\Theta\rfloor; \lfloor\Gamma\rfloor \vdash \texttt{let}\, f : \lfloor A \to B\rfloor = c;\ \texttt{let}\, x : \lfloor A\rfloor = c';\ \texttt{let}\, y : \lfloor B\rfloor = f(x);\ \texttt{return}\, y : \uparrow\lfloor B\rfloor$, we gradually apply the corresponding rules of $\text{F}^{\pm}_{\exists}$ to construct the required typing inference tree:

1. $\lfloor\Theta\rfloor; \lfloor\Gamma\rfloor, f : \downarrow(\lfloor A\rfloor \to \uparrow\lfloor B\rfloor), x : \lfloor A\rfloor, y : \lfloor B\rfloor \vdash \texttt{return}\, y : \uparrow\lfloor B\rfloor$ by (VAR^{INF}) and (RET^{INF}).
2. $\lfloor\Theta\rfloor; \lfloor\Gamma\rfloor, f : \downarrow(\lfloor A\rfloor \to \uparrow\lfloor B\rfloor), x : \lfloor A\rfloor \vdash \texttt{let}\, y : \lfloor B\rfloor = f(x);\ \texttt{return}\, y : \uparrow\lfloor B\rfloor$ by ($\text{LET}^{\text{INF}}_{:@}$). Let us show that the required premises hold.
   a. $\lfloor\Theta\rfloor \vdash \lfloor B\rfloor$ by **??**;
   b. $\lfloor\Theta\rfloor; \lfloor\Gamma\rfloor, f : \downarrow(\lfloor A\rfloor \to \uparrow\lfloor B\rfloor), x : \lfloor A\rfloor \vdash f : \downarrow(\lfloor A\rfloor \to \uparrow\lfloor B\rfloor)$ by (VAR^{INF});

c. $\lfloor\Theta\rfloor\,;\lfloor\Gamma\rfloor, f:\downarrow(\lfloor A\rfloor\to\uparrow\lfloor B\rfloor), x:\lfloor A\rfloor\vdash(\lfloor A\rfloor\to\uparrow\lfloor B\rfloor)\bullet x\Rrightarrow\uparrow\lfloor B\rfloor$ by $(\to^{\mathrm{INF}}_{\bullet\Rightarrow})$;

d. $\lfloor\Theta\rfloor\vdash\uparrow\lfloor B\rfloor\leqslant\uparrow\lfloor B\rfloor$ by reflexivity (Lemma 22);

e. $\lfloor\Theta\rfloor;\lfloor\Gamma\rfloor, f:\downarrow(\lfloor A\rfloor\to\uparrow\lfloor B\rfloor), x:\lfloor A\rfloor, y:\lfloor B\rfloor\vdash\mathtt{return}\,y:\uparrow\lfloor B\rfloor$ as noted above;

3. $\lfloor\Theta\rfloor;\lfloor\Gamma\rfloor, f:\downarrow(\lfloor A\rfloor\to\uparrow\lfloor B\rfloor)\vdash\mathtt{let}\,x:\lfloor A\rfloor=c';\ \mathtt{let}\,y:\lfloor B\rfloor=f(x);\ \mathtt{return}\,y:\uparrow\lfloor B\rfloor$ by $(\mathrm{LET}^{\mathrm{INF}}_{\mathrm{C}})$. Let us show that the required premises hold.

a. $\lfloor\Theta\rfloor\vdash\lfloor A\rfloor$ by **??**;

b. $\lfloor\Theta\rfloor;\lfloor\Gamma\rfloor, f:\downarrow(\lfloor A\rfloor\to\uparrow\lfloor B\rfloor)\vdash c':\uparrow\lfloor A\rfloor$ holds by weakening of $\lfloor\Theta\rfloor;\lfloor\Gamma\rfloor\vdash c':\uparrow\lfloor A\rfloor$, which in turn holds by the induction hypothesis, as noted above;

c. $\lfloor\Theta\rfloor\vdash\uparrow\lfloor A\rfloor\leqslant\uparrow\lfloor A\rfloor$ by reflexivity (Lemma 22);

d. $\lfloor\Theta\rfloor;\lfloor\Gamma\rfloor, f:\downarrow(\lfloor A\rfloor\to\uparrow\lfloor B\rfloor), x:\lfloor A\rfloor\vdash\mathtt{let}\,y:\lfloor B\rfloor=f(x);\ \mathtt{return}\,y:\uparrow\lfloor B\rfloor$ as noted above;

4. $\lfloor\Theta\rfloor;\lfloor\Gamma\rfloor\vdash\mathtt{let}\,f:\lfloor A\to B\rfloor=c;\ \mathtt{let}\,x:\lfloor A\rfloor=c';\ \mathtt{let}\,y:\lfloor B\rfloor=f(x);\ \mathtt{return}\,y:\uparrow\lfloor B\rfloor$ by $(\mathrm{LET}^{\mathrm{INF}}_{\mathrm{C}})$. Let us show that the required premises hold.

a. $\lfloor\Theta\rfloor\vdash\lfloor A\to B\rfloor$ by **??**;

b. $\lfloor\Theta\rfloor;\lfloor\Gamma\rfloor\vdash c:\uparrow\lfloor A\to B\rfloor$ by the induction hypothesis, as noted above;

c. $\lfloor\Theta\rfloor\vdash\uparrow\lfloor A\to B\rfloor\leqslant\uparrow\lfloor A\to B\rfloor$ by reflexivity (Lemma 22);

d. $\lfloor\Theta\rfloor;\lfloor\Gamma\rfloor, f:\downarrow(\lfloor A\rfloor\to\uparrow\lfloor B\rfloor)\vdash\mathtt{let}\,x:\lfloor A\rfloor=c';\ \mathtt{let}\,y:\lfloor B\rfloor=f(x);\ \mathtt{return}\,y:\uparrow\lfloor B\rfloor$ as noted above (we rewrote $\lfloor A\to B\rfloor$ as $\downarrow(\lfloor A\rfloor\to\uparrow\lfloor B\rfloor)$).

Case 5. $(\mathsf{TApp}^{\rightsquigarrow^{\pm}})$ By inversion of this rule, we have

- $\Theta;\Gamma\vdash t:\forall\alpha.T\rightsquigarrow^{\pm}c$, and thus, $\lfloor\Theta\rfloor;\lfloor\Gamma\rfloor\vdash c:\uparrow\lfloor\forall\alpha.T\rfloor$ by the induction hypothesis; and
- $\Theta\vdash A$, which implies $\lfloor\Theta\rfloor\vdash\lfloor A\rfloor$ by **??**.

We wish to infer $\lfloor\Theta\rfloor;\lfloor\Gamma\rfloor\vdash\mathtt{let}\,f:\lfloor\forall\alpha.T\rfloor=c;\ \mathtt{let}\,y:\lfloor[A/\alpha]T\rfloor=f();\ \mathtt{return}\,y:\uparrow\lfloor[A/\alpha]T\rfloor$. First, let us rewrite the polarized types by definition and Lemma 54: $\lfloor\Theta\rfloor;\lfloor\Gamma\rfloor\vdash\mathtt{let}\,f:\downarrow\forall\alpha^{+}.\uparrow\lfloor T\rfloor=c;\ \mathtt{let}\,y:[\lfloor A\rfloor/\alpha^{+}]\lfloor T\rfloor=f();\ \mathtt{return}\,y:\uparrow[\lfloor A\rfloor/\alpha^{+}]\lfloor T\rfloor$. Then we prove this judgment gradually by applying the corresponding rules of $\mathrm{F}^{\pm}_{\exists}$:

1. $\lfloor\Theta\rfloor;\lfloor\Gamma\rfloor, f:\downarrow\forall\alpha^{+}.\uparrow\lfloor T\rfloor, y:[\lfloor A\rfloor/\alpha^{+}]\lfloor T\rfloor\vdash\mathtt{return}\,y:\uparrow[\lfloor A\rfloor/\alpha^{+}]\lfloor T\rfloor$ holds by $(\mathrm{VAR}^{\mathrm{INF}})$ and $(\mathrm{RET}^{\mathrm{INF}})$.
2. $\lfloor\Theta\rfloor;\lfloor\Gamma\rfloor, f:\downarrow\forall\alpha^{+}.\uparrow\lfloor T\rfloor\vdash\mathtt{let}\,y:[\lfloor A\rfloor/\alpha^{+}]\lfloor T\rfloor=f();\ \mathtt{return}\,y:\uparrow[\lfloor A\rfloor/\alpha^{+}]\lfloor T\rfloor$ holds by $(\mathrm{LET}^{\mathrm{INF}}_{:@})$. Let us show that the required premises hold.

a. $\lfloor\Theta\rfloor \vdash [\lfloor A\rfloor/\alpha^+]\lfloor T\rfloor$ holds since $\lfloor\Theta\rfloor \vdash \lfloor A\rfloor$ and $\lfloor\Theta\rfloor \vdash \lfloor T\rfloor$ by Lemma 5.

b. $\lfloor\Theta\rfloor; \lfloor\Gamma\rfloor, f : \downarrow\forall\alpha^+.\uparrow\lfloor T\rfloor \vdash f : \downarrow\forall\alpha^+.\uparrow\lfloor T\rfloor$ by (VAR^{INF}).

c. $\lfloor\Theta\rfloor; \lfloor\Gamma\rfloor, f : \downarrow\forall\alpha^+.\uparrow\lfloor T\rfloor \vdash \forall\alpha^+.\uparrow\lfloor T\rfloor \bullet \Longrightarrow \forall\alpha^+.\uparrow\lfloor T\rfloor$ by ($\emptyset^{\text{INF}}_{\bullet\Longrightarrow}$).

d. $\lfloor\Theta\rfloor \vdash \forall\alpha^+.\uparrow\lfloor T\rfloor \leqslant \uparrow[\lfloor A\rfloor/\alpha^+]\lfloor T\rfloor$ by ($\forall^{\leqslant}$): $\lfloor\Theta\rfloor \vdash \lfloor A\rfloor/\alpha^+ : \alpha^+$ since $\lfloor\Theta\rfloor \vdash \lfloor A\rfloor$, as noted above.

e. $\lfloor\Theta\rfloor; \lfloor\Gamma\rfloor, f : \downarrow\forall\alpha^+.\uparrow\lfloor T\rfloor, y : [\lfloor A\rfloor/\alpha^+]\lfloor T\rfloor \vdash \texttt{return}\, y : \uparrow[\lfloor A\rfloor/\alpha^+]\lfloor T\rfloor$ as noted above.

3. $\lfloor\Theta\rfloor; \lfloor\Gamma\rfloor \vdash \texttt{let}\, f : \downarrow\forall\alpha^+.\uparrow\lfloor T\rfloor = c;\ \texttt{let}\, y : [\lfloor A\rfloor/\alpha^+]\lfloor T\rfloor = f();\ \texttt{return}\, y : \uparrow[\lfloor A\rfloor/\alpha^+]\lfloor T\rfloor$ holds by ($\text{LET}^{\text{INF}}_{\text{C}}$). Let us show that the required premises hold.

a. $\lfloor\Theta\rfloor \vdash \downarrow\forall\alpha^+.\uparrow\lfloor T\rfloor$ by **??**.

b. $\lfloor\Theta\rfloor; \lfloor\Gamma\rfloor \vdash c : \uparrow\downarrow\forall\alpha^+.\uparrow\lfloor T\rfloor$ by the induction hypothesis, as noted above.

c. $\lfloor\Theta\rfloor \vdash \uparrow\downarrow\forall\alpha^+.\uparrow\lfloor T\rfloor \leqslant \uparrow\downarrow\forall\alpha^+.\uparrow\lfloor T\rfloor$ by reflexivity (Lemma 22).

d. $\lfloor\Theta\rfloor; \lfloor\Gamma\rfloor, f : \downarrow\forall\alpha^+.\uparrow\lfloor T\rfloor \vdash \texttt{let}\, y : [\lfloor A\rfloor/\alpha^+]\lfloor T\rfloor = f();\ \texttt{return}\, y : \uparrow[\lfloor A\rfloor/\alpha^+]\lfloor T\rfloor$ as noted above.

■

C.3 Algorithmic Types

C.3.1 Algorithmic Type Well-formedness

Lemma 58 (Soundness of algorithmic type well-formedness)**.**

+ *if* $\Theta\,;\widehat{\Theta} \vdash P$ *then* $\texttt{fv}\,(P) \subseteq \Theta$ *and* $\texttt{fav}(P) \subseteq \widehat{\Theta}$*;*
− *if* $\Theta\,;\widehat{\Theta} \vdash N$ *then* $\texttt{fv}\,(N) \subseteq \Theta$ *and* $\texttt{fav}(N) \subseteq \widehat{\Theta}$.

Proof The proof is analogous to Lemma 3. The additional base case is when $\Theta\,;\widehat{\Theta} \vdash P$ is derived by ($\text{UVAR}^{\text{WF}}_{+}$), and the symmetric negative case. In this case, $P = \widehat{\alpha}^+$, and $\texttt{fav}(P) = \widehat{\alpha}^+ \subseteq \widehat{\Theta}$ by inversion; $\texttt{fv}\,(P) = \emptyset \subseteq \Theta$ vacuously. ■

Lemma 59 (Completeness of algorithmic type well-formedness)**.** *In the well-formedness judgment, only used variables matter:*

+ *if* $\Theta_1 \cap \texttt{fv}\, P = \Theta_2 \cap \texttt{fv}\, P$ *and* $\widehat{\Theta}_1 \cap \texttt{fav} P = \widehat{\Theta}_2 \cap \texttt{fav} P$ *then* $\Theta_1\,;\widehat{\Theta}_1 \vdash P \iff \Theta_2\,;\widehat{\Theta}_2 \vdash P$*, and*
− *if* $\Theta_1 \cap \texttt{fv}\, N = \Theta_2 \cap \texttt{fv}\, N$ *and* $\widehat{\Theta}_1 \cap \texttt{fav} N = \widehat{\Theta}_2 \cap \texttt{fav} N$ *then* $\Theta_1\,;\widehat{\Theta}_1 \vdash N \iff \Theta_2\,;\widehat{\Theta}_2 \vdash N$.

Proof By mutual structural induction on P and N. ■

Lemma 60 (Variable algorithmization agrees with well-formedness)**.**

+ $\Theta, \overrightarrow{\alpha^-} \vdash P$ *implies* $\Theta\,;\overrightarrow{\widehat{\alpha}^-} \vdash [\overrightarrow{\widehat{\alpha}^-}/\overrightarrow{\alpha^-}]P$*;*

– $\Theta, \overrightarrow{\alpha^-} \vdash N$ *implies* $\Theta ; \overrightarrow{\widehat{\alpha}^-} \vdash [\overrightarrow{\widehat{\alpha}^-}/\overrightarrow{\alpha^-}]N$.

Proof The proof is a structural induction on $\Theta, \overrightarrow{\alpha^-} \vdash P$ and mutually, on $\Theta, \overrightarrow{\alpha^-} \vdash N$. Notice that the substitutions commute with all the constructors, providing the step of the induction. ■

Lemma 61 (Variable de-algorithmization agrees with well-formedness)**.**

+ $\Theta ; \overrightarrow{\widehat{\alpha}^-} \vdash P$ *implies* $\Theta, \overrightarrow{\alpha^-} \vdash [\overrightarrow{\alpha^-}/\overrightarrow{\widehat{\alpha}^-}]P$;
– $\Theta ; \overrightarrow{\widehat{\alpha}^-} \vdash N$ *implies* $\Theta, \overrightarrow{\alpha^-} \vdash [\overrightarrow{\alpha^-}/\overrightarrow{\widehat{\alpha}^-}]N$.

Proof As for Lemma 60, the proof is a structural induction on $\Theta ; \overrightarrow{\widehat{\alpha}^-} \vdash P$ and mutually, on $\Theta ; \overrightarrow{\widehat{\alpha}^-} \vdash N$. ■

Corollary 22 (Well-formedness Algorithmic Context Weakening)**.** *Suppose that* $\Theta_1 \subseteq \Theta_2$, *and* $\widehat{\Theta}_1 \subseteq \widehat{\Theta}_2$. *Then*

+ *if* $\Theta_1 ; \widehat{\Theta}_1 \vdash P$ *implies* $\Theta_2 ; \widehat{\Theta}_2 \vdash P$,
– *if* $\Theta_1 ; \widehat{\Theta}_1 \vdash N$ *implies* $\Theta_2 ; \widehat{\Theta}_2 \vdash N$.

Proof By Lemma 58, $\Theta_1 ; \widehat{\Theta}_1 \vdash P$ implies $\mathtt{fv}(P) \subseteq \Theta_1 \subseteq \Theta_2$ and $\mathtt{fav}(P) \subseteq \widehat{\Theta}_1 \subseteq \widehat{\Theta}_2$, and thus, $\mathtt{fv}(P) = \mathtt{fv}(P) \cap \Theta_1 = \mathtt{fv}(P) \cap \Theta_2$, and $\mathtt{fav}(P) = \mathtt{fav}(P) \cap \widehat{\Theta}_1 = \mathtt{fav}(P) \cap \widehat{\Theta}_2$. Then by Lemma 59, $\Theta_2 ; \widehat{\Theta}_2 \vdash P$. The negative case is symmetric. ■

C.3.2 Substitution

Lemma 63 (Algorithmic Substitution Strengthening)**.** *Restricting the substitution to the algorithmic variables of the substitution subject does not affect the result. Suppose that* $\widehat{\sigma}$ *is an algorithmic substitution,* P *and* N *are algorithmic types. Then*

+ $[\widehat{\sigma}]P = [\widehat{\sigma}|_{\mathtt{fav}P}]P$,
– $[\widehat{\sigma}]N = [\widehat{\sigma}|_{\mathtt{fav}N}]N$

Proof The proof is analogous to the proof of Lemma 6. ■

Lemma 64 (Substitutions equal on the algorithmic variables)**.** *Suppose that* $\widehat{\sigma}_1$ *and* $\widehat{\sigma}_2$ *are normalized substitutions of signature* $\Xi \vdash \widehat{\sigma}_i : \widehat{\Theta}$. *Then*

+ *for a normalized type* $\Theta ; \widehat{\Theta} \vdash P$, *if* $[\widehat{\sigma}_1]P = [\widehat{\sigma}_2]P$ *then* $\widehat{\sigma}_1|_{(\mathtt{fav}P)} = \widehat{\sigma}_2|_{(\mathtt{fav}P)}$;
– *for a normalized type* $\Theta ; \widehat{\Theta} \vdash N$, *if* $[\widehat{\sigma}_1]N = [\widehat{\sigma}_2]N$ *then* $\widehat{\sigma}_1|_{(\mathtt{fav}N)} = \widehat{\sigma}_2|_{(\mathtt{fav}N)}$.

Proof The proof is a simple structural induction on $\Theta ; \widehat{\Theta} \vdash P$ and mutually, on $\Theta ; \widehat{\Theta} \vdash N$. Let us consider the shape of N (the cases of P are symmetric).

Case 1. $\Theta ; \widehat{\Theta} \vdash \widehat{\alpha}^-$. Then $[\widehat{\sigma}_1]\widehat{\alpha}^- = [\widehat{\sigma}_2]\widehat{\alpha}^-$ implies $\widehat{\sigma}_1|_{(\mathtt{fav}\widehat{\alpha}^-)} = \widehat{\sigma}_2|_{(\mathtt{fav}\widehat{\alpha}^-)}$ immediately.

Case 2. $\Theta ; \widehat{\Theta} \vdash \alpha^-$. Then $\mathtt{fav}\widehat{\alpha}^- = \emptyset$, and $\widehat{\sigma}_1|_{(\mathtt{fav}\widehat{\alpha}^-)} = \widehat{\sigma}_2|_{(\mathtt{fav}\widehat{\alpha}^-)}$ holds vacuously.

Case 3. $\Theta ; \widehat{\Theta} \vdash \forall \overrightarrow{\alpha^{\mp}}. N$. Then we are proving that $[\widehat{\sigma_1}]\forall \overrightarrow{\alpha^{\mp}}. N = [\widehat{\sigma_2}]\forall \overrightarrow{\alpha^{\mp}}. N$ implies $\widehat{\sigma_1}|_{(\mathtt{fav}\forall \overrightarrow{\alpha^{\mp}}. N)} = \widehat{\sigma_2}|_{(\mathtt{fav}\forall \overrightarrow{\alpha^{\mp}}. N)}$. By definition of substitution and $(\forall^{\simeq^D})$, $[\widehat{\sigma_1}]N = [\widehat{\sigma_2}]N$ implies $\widehat{\sigma_1}|_{\mathtt{fav}N} = \widehat{\sigma_2}|_{\mathtt{fav}N}$. Since $\forall \overrightarrow{\alpha^{\mp}}. N$ is normalized, so is $\Theta, \overrightarrow{\alpha^{\mp}} ; \widehat{\Theta} \vdash N$, hence, the induction hypothesis is applicable and implies $\widehat{\sigma_1}|_{\mathtt{fav}N} = \widehat{\sigma_2}|_{\mathtt{fav}N}$, as required.

Case 4. $\Theta ; \widehat{\Theta} \vdash P \to N$. Then we are proving that $[\widehat{\sigma_1}](P \to N) = [\widehat{\sigma_2}](P \to N)$ implies $\widehat{\sigma_1}|_{(\mathtt{fav}P \to N)} = \widehat{\sigma_2}|_{(\mathtt{fav}P \to N)}$. By definition of substitution and congruence of equality, $[\widehat{\sigma_1}](P \to N) = [\widehat{\sigma_2}](P \to N)$ means $[\widehat{\sigma_1}]P = [\widehat{\sigma_2}]P$ and $[\widehat{\sigma_1}]N = [\widehat{\sigma_2}]N$. Notice that P and N are normalized since $P \to N$ is normalized, and well-formed in the same contexts. This way, by the induction hypothesis, $\widehat{\sigma_1}|_{(\mathtt{fav}P)} = \widehat{\sigma_2}|_{(\mathtt{fav}P)}$ and $\widehat{\sigma_1}|_{(\mathtt{fav}N)} = \widehat{\sigma_2}|_{(\mathtt{fav}N)}$, which since $\mathtt{fav}(P \to N) = \mathtt{fav}P \cup \mathtt{fav}N$ implies $\widehat{\sigma_1}|_{(\mathtt{fav}P \to N)} = \widehat{\sigma_2}|_{(\mathtt{fav}P \to N)}$.

Case 5. $\Theta ; \widehat{\Theta} \vdash \uparrow P$. The proof is similar to the previous case: we apply congruence of substitution, equality, and normalization, then the induction hypothesis, and then the fact that $\mathtt{fav}(\uparrow P) = \mathtt{fav}P$.

■

Corollary 23 (Substitutions equivalent on the algorithmic variables)**.** *Suppose that* $\widehat{\sigma_1}$ *and* $\widehat{\sigma_2}$ *are substitutions of signature* $\Xi \vdash \widehat{\sigma_i} : \widehat{\Theta}$ *where* $\Theta \vdash^{\supseteq} \Xi$*. Then*

+ *for a type* $\Theta ; \widehat{\Theta} \vdash P$*, if* $\Theta \vdash [\widehat{\sigma_1}]P \simeq^{\leqslant} [\widehat{\sigma_2}]P$ *then* $\Xi \vdash \widehat{\sigma_1} \simeq^{\leqslant} \widehat{\sigma_2} : \mathtt{fav}P$*;*
− *for a type* $\Theta ; \widehat{\Theta} \vdash N$*, if* $\Theta \vdash [\widehat{\sigma_1}]N \simeq^{\leqslant} [\widehat{\sigma_2}]N$ *then* $\Xi \vdash \widehat{\sigma_1} \simeq^{\leqslant} \widehat{\sigma_2} : \mathtt{fav}N$*.*

Proof First, let us normalize the types and the substitutions, and show that the given equivalences and well-formedness properties are preserved. $\Theta ; \widehat{\Theta} \vdash P$ implies $\Theta ; \widehat{\Theta} \vdash \mathtt{nf}\,(P)$ by Corollary 24. $\Xi \vdash [\widehat{\sigma_1}]P \simeq^D [\widehat{\sigma_2}]P$ implies $\mathtt{nf}\,([\widehat{\sigma_1}]P) = \mathtt{nf}\,([\widehat{\sigma_2}]P)$ by Lemma 48. Then $\mathtt{nf}\,([\widehat{\sigma_1}]P) = \mathtt{nf}\,([\widehat{\sigma_2}]P)$ implies $[\mathtt{nf}\,(\widehat{\sigma_1})]\mathtt{nf}\,(P) = [\mathtt{nf}\,(\widehat{\sigma_2})]\mathtt{nf}\,(P)$ by Lemma 43. Notice that by Corollary 25 $\Xi \vdash \widehat{\sigma_i} : \widehat{\Theta}$ implies $\Xi \vdash \mathtt{nf}\,(\widehat{\sigma_i}) : \widehat{\Theta}$.

This way, by Lemma 64, $\Xi \vdash [\widehat{\sigma_1}]P \simeq^D [\widehat{\sigma_2}]P$ implies $\mathtt{nf}\,(\widehat{\sigma_1})|_{(\mathtt{favnf}\,(P))} = \mathtt{nf}\,(\widehat{\sigma_2})|_{(\mathtt{favnf}\,(P))}$. Then by Lemma 66, $\mathtt{nf}\,(\widehat{\sigma_1})|_{(\mathtt{fav}P)} = \mathtt{nf}\,(\widehat{\sigma_2})|_{(\mathtt{fav}P)}$, and by Corollary 26, $\Xi \vdash \widehat{\sigma_1} \simeq^{\leqslant} \widehat{\sigma_2} : \mathtt{fav}P$.

Symmetrically, $\Xi \vdash [\widehat{\sigma_1}]N \simeq^D [\widehat{\sigma_2}]N$ implies $\Xi \vdash \widehat{\sigma_1} \simeq^{\leqslant} \widehat{\sigma_2} : \mathtt{fav}N$. ■

C.3.3 Normalization

Lemma 66 (Algorithmic variables are not changed by the normalization)**.**

− $\mathtt{fav}N \equiv \mathtt{favnf}\,(N)$
+ $\mathtt{fav}P \equiv \mathtt{favnf}\,(P)$

Proof By straightforward induction on N and mutually on P, similar to the proof of Lemma 40. ■

Lemma 67 (Soundness of normalization of algorithmic types)**.**

- $N \simeq^D \mathtt{nf}\,(N)$
- $P \simeq^D \mathtt{nf}\,(P)$

Proof The proof coincides with the proof of Lemma 41. ∎

C.3.4 Equivalence

Lemma 68 (Algorithmic type well-formedness is invariant under equivalence)**.** *Mutual subtyping implies declarative equivalence.*

- *if* $P \simeq^D Q$ *then* $\Theta\,;\widehat{\Theta} \vdash P \iff \Theta\,;\widehat{\Theta} \vdash Q$,
- *if* $N \simeq^D M$ *then* $\Theta\,;\widehat{\Theta} \vdash N \iff \Theta\,;\widehat{\Theta} \vdash M$

Proof The proof coincides with the proof of Lemma 28, and adds two cases for equating two positive or two negative algorithmic variables, which must be equal by inversion, and thus, $\Theta\,;\widehat{\Theta} \vdash \widehat{\alpha}^{\pm} \iff \Theta\,;\widehat{\Theta} \vdash \widehat{\alpha}^{\pm}$ holds trivially. ∎

Corollary 24 (Normalization preserves well-formedness of algorithmic types)**.**

- $\Theta\,;\widehat{\Theta} \vdash P \iff \Theta\,;\widehat{\Theta} \vdash \mathtt{nf}\,(P)$,
- $\Theta\,;\widehat{\Theta} \vdash N \iff \Theta\,;\widehat{\Theta} \vdash \mathtt{nf}\,(N)$

Proof Immediately from Lemmas 67 and 68. ∎

Corollary 25 (Normalization preserves the signature of the algorithmic substitution)**.** $\Xi \vdash \widehat{\sigma} : \widehat{\Theta} \iff \Xi \vdash \mathtt{nf}\,(\widehat{\sigma}) : \widehat{\Theta}$, $\Theta \vdash \widehat{\sigma} : \widehat{\Theta} \iff \Theta \vdash \mathtt{nf}\,(\widehat{\sigma}) : \widehat{\Theta}$.

Proof The proof is analogous to Corollary 15. ∎

Corollary 26 (Algorithmic substitution equivalence becomes equality after normalization)**.** *Suppose that* $\Xi \vdash \widehat{\sigma}_1 : \widehat{\Theta}'$ *and* $\Xi \vdash \widehat{\sigma}_2 : \widehat{\Theta}'$ *are algorithmic substitutions and* $\widehat{\Theta} \subseteq \widehat{\Theta}'$*. Then* $\Xi \vdash \widehat{\sigma}_1 \simeq^{\leqslant} \widehat{\sigma}_2 : \widehat{\Theta} \iff \mathtt{nf}\,(\widehat{\sigma}_1)|_{\widehat{\Theta}} = \mathtt{nf}\,(\widehat{\sigma}_2)|_{\widehat{\Theta}}$.

Proof Follows immediately from Lemma 48:

- ⇒ If $\widehat{\alpha}^{\pm} \notin \widehat{\Theta}$, then $[\mathtt{nf}\,(\widehat{\sigma}_1)|_{\widehat{\Theta}}]\widehat{\alpha}^{\pm} = [\mathtt{nf}\,(\widehat{\sigma}_2)|_{\widehat{\Theta}}]\widehat{\alpha}^{\pm} = \widehat{\alpha}^{\pm}$ by definition. For any $\widehat{\alpha}^{\pm} \in \widehat{\Theta}$, $[\mathtt{nf}\,(\widehat{\sigma}_1)|_{\widehat{\Theta}}]\widehat{\alpha}^{\pm} = \mathtt{nf}\,([\widehat{\sigma}_1]\widehat{\alpha}^{\pm})$ and $[\mathtt{nf}\,(\widehat{\sigma}_2)|_{\widehat{\Theta}}]\widehat{\alpha}^{\pm} = \mathtt{nf}\,([\widehat{\sigma}_2]\widehat{\alpha}^{\pm})$; $\Xi(\widehat{\alpha}^{\pm}) \vdash [\widehat{\sigma}_1]\widehat{\alpha}^{\pm} \simeq^{\leqslant} [\widehat{\sigma}_2]\widehat{\alpha}^{\pm}$ implies $\mathtt{nf}\,([\widehat{\sigma}_1]\widehat{\alpha}^{\pm}) = \mathtt{nf}\,([\widehat{\sigma}_2]\widehat{\alpha}^{\pm})$ by Lemma 45.
- ⇐ If $\widehat{\alpha}^{\pm} \in \widehat{\Theta}$, then $\mathtt{nf}\,(\widehat{\sigma}_1)|_{\widehat{\Theta}} = \mathtt{nf}\,(\widehat{\sigma}_2)|_{\widehat{\Theta}}$ implies $\mathtt{nf}\,([\widehat{\sigma}_1]\widehat{\alpha}^{\pm}) = \mathtt{nf}\,([\widehat{\sigma}_2]\widehat{\alpha}^{\pm})$ by definition of substitution restriction and normalization. In turn, $\mathtt{nf}\,([\widehat{\sigma}_1]\widehat{\alpha}^{\pm}) = \mathtt{nf}\,([\widehat{\sigma}_2]\widehat{\alpha}^{\pm})$ means $\Xi(\widehat{\alpha}^{\pm}) \vdash [\widehat{\sigma}_1]\widehat{\alpha}^{\pm} \simeq^{\leqslant} [\widehat{\sigma}_2]\widehat{\alpha}^{\pm}$ by Lemma 45.

∎

C.3.5 Unification Constraint Merge

Observation 6 (Unification Constraint Merge Determinism)**.** *Suppose that* $\Xi \vdash UC_1$ *and* $\Xi \vdash UC_2$ *If* $\Xi \vdash UC_1 \mathbin{\&} UC_2 = UC$ *and* $\Xi \vdash UC_1 \mathbin{\&} UC_2 = UC'$ *are defined then* $UC = UC'$.

Proof UC and UC' both consists of three parts: Entries of UC_1 that do not have matching entries in UC_2, entries of UC_2 that do not have matching entries in UC_1, and the merge of matching entries.

The parts corresponding to unmatched entries of UC_1 and UC_2 coincide, since UC_1 and UC_2 are fixed. To show that the merge of matching entries coincide, let us take any pair of matching $ue_1 \in UC_1$ and $ue_2 \in UC_2$ and consider their shape.

Case 1. ue_1 is $\widehat{\alpha}^+ :\simeq Q_1$ and ue_2 is $\widehat{\alpha}^+ :\simeq Q_2$ then the result, if it exists, is always ue_1, by inversion of $(\simeq \&^+ \simeq)$.

Case 2. ue_1 is $\widehat{\alpha}^- :\simeq N_1$ and ue_2 is $\widehat{\alpha}^- :\simeq N_2$ then analogously, the result, if it exists, is always ue_1, by inversion of $(\simeq \&^- \simeq)$.

This way, the third group of entries coincide as well. ∎

Lemma 69 (Soundness of Unification Constraint Merge)**.** *Suppose that* $\Xi \vdash UC_1$ *and* $\Xi \vdash UC_2$ *are normalized unification constraints. If* $\Xi \vdash UC_1 \mathbin{\&} UC_2 = UC$ *is defined then* $UC = UC_1 \cup UC_2$.

Proof

- $UC_1 \mathbin{\&} UC_2 \subseteq UC_1 \cup UC_2$
  By definition, $UC_1 \mathbin{\&} UC_2$ consists of three parts: entries of UC_1 that do not have matching entries of UC_2, entries of UC_2 that do not have matching entries of UC_1, and the merge of matching entries.

  If ue is from the first or the second part, then $ue \in UC_1 \cup UC_2$ holds immediately. If ue is from the third part, then ue is the merge of two matching entries $ue_1 \in UC_1$ and $ue_2 \in UC_2$. Since UC_1 and UC_2 are normalized unification , ue_1 and ue_2 have one of the following forms:

  - $\widehat{\alpha}^+ :\simeq P_1$ and $\widehat{\alpha}^+ :\simeq P_2$, where P_1 and P_2 are normalized, and then since $\Xi(\widehat{\alpha}^+) \vdash ue_1 \mathbin{\&} ue_2 = ue$ exists, $(\simeq \&^+ \simeq)$ was applied to infer it. It means that $ue = ue_1 = ue_2$;
  - $\widehat{\alpha}^- :\simeq N_1$ and $\widehat{\alpha}^- :\simeq N_2$, then symmetrically, $\Xi(\widehat{\alpha}^-) \vdash ue_1 \mathbin{\&} ue_2 = ue = ue_1 = ue_2$

  In both cases, $ue \in UC_1 \cup UC_2$.

- $UC_1 \cup UC_2 \subseteq UC_1 \mathbin{\&} UC_2$
  Let us take an arbitrary $ue_1 \in UC_1$. Then since UC_1 is a unification constraint, ue_1 has one of the following forms:

  - $\widehat{\alpha}^+ :\simeq P$ where P is normalized. If $\widehat{\alpha}^+ \notin \mathtt{dom}\,(UC_2)$, then $ue_1 \in UC_1 \mathbin{\&} UC_2$. Otherwise, there is a normalized matching $ue_2 = (\widehat{\alpha}^+ :\simeq P') \in UC_2$ and then

since UC_1 & UC_2 exists, $(\simeq \&^+ \simeq)$ was applied to construct ue_1 & $ue_2 \in UC_1$ & UC_2. By inversion of $(\simeq \&^+ \simeq)$, ue_1 & $ue_2 = ue_1$, and $\mathtt{nf}(P) = \mathtt{nf}(P')$, which since P and P' are normalized, implies that $P = P'$, that is $ue_1 = ue_2 \in UC_1$ & UC_2.

– $\widehat{\alpha}^- :\simeq N$ where N is normalized. Then symmetrically, $ue_1 = ue_2 \in UC_1$ & UC_2.

Similarly, if we take an arbitrary $ue_2 \in UC_2$, then $ue_1 = ue_2 \in UC_1$ & UC_2.

■

Corollary 27. *Suppose that $\Xi \vdash UC_1$ and $\Xi \vdash UC_2$ are normalized unification constraints. If $\Xi \vdash UC_1$ & $UC_2 = UC$ is defined then*

1. *$\Xi \vdash UC$ is normalized unification constraint,*
2. *for any substitution $\Xi \vdash \widehat{\sigma} : \mathtt{dom}(UC)$, $\Xi \vdash \widehat{\sigma} : UC$ implies $\Xi \vdash \widehat{\sigma} : UC_1$ and $\Xi \vdash \widehat{\sigma} : UC_2$.*

Proof It is clear that since $UC = UC_1 \cup UC_2$ (by Lemma 69), and being normalized means that all entries are normalized, UC is a normalized unification constraint. Analogously, $\Xi \vdash UC = UC_1 \cup UC_2$ holds immediately, since $\Xi \vdash UC_1$ and $\Xi \vdash UC_2$.

Let us take an arbitrary substitution $\Xi \vdash \widehat{\sigma} : \mathtt{dom}(UC)$ and assume that $\Xi \vdash \widehat{\sigma} : UC$. Then $\Xi \vdash \widehat{\sigma} : UC_i$ holds by definition: If $ue \in UC_i \subseteq UC_1 \cup UC_2 = UC$ then $\Xi(\widehat{\alpha}^{\pm}) \vdash [\widehat{\sigma}]\widehat{\alpha}^{\pm} : ue$ (where ue restricts $\widehat{\alpha}^{\pm}$) holds since $\Xi \vdash \widehat{\sigma} : \mathtt{dom}(UC)$. ■

Lemma 70 (Completeness of Unification Constraint Entry Merge)**.** *For a fixed context Θ, suppose that $\Theta \vdash ue_1$ and $\Theta \vdash ue_2$ are matching constraint entries.*

+ *for a type P such that $\Theta \vdash P : ue_1$ and $\Theta \vdash P : ue_2$, $\Theta \vdash ue_1$ & $ue_2 = ue$ is defined and $\Theta \vdash P : ue$.*
– *for a type N such that $\Theta \vdash N : ue_1$ and $\Theta \vdash N : ue_2$, $\Theta \vdash ue_1$ & $ue_2 = ue$ is defined and $\Theta \vdash N : ue$.*

Proof Let us consider the shape of ue_1 and ue_2.

Case 1. ue_1 is $\widehat{\alpha}^+ :\simeq Q_1$ and ue_2 is $\widehat{\alpha}^+ :\simeq Q_2$. Then $\Theta \vdash P : ue_1$ means $\Theta \vdash P \simeq^{\leqslant} Q_1$, and $\Theta \vdash P : ue_2$ means $\Theta \vdash P \simeq^{\leqslant} Q_2$. Then by transitivity of equivalence (Corollary 10), $\Theta \vdash Q_1 \simeq^{\leqslant} Q_2$, which means $\mathtt{nf}(Q_1) = \mathtt{nf}(Q_2)$ by Lemma 48. Hence, $(\simeq \&^+ \simeq)$ applies to infer $\Theta \vdash ue_1$ & $ue_2 = ue_2$, and $\Theta \vdash P : ue_2$ holds by assumption.

Case 2. ue_1 is $\widehat{\alpha}^- :\simeq N_1$ and ue_2 is $\widehat{\alpha}^- :\simeq M_2$. The proof is symmetric.

■

Lemma 71 (Completeness of Unification Constraint Merge)**.** *Suppose that $\Xi \vdash UC_1$ and $\Xi \vdash UC_2$. Then for any $\widehat{\Theta} \supseteq \mathtt{dom}(UC_1) \cup \mathtt{dom}(UC_2)$ and substitution $\Xi \vdash \widehat{\sigma} : \widehat{\Theta}$ such that $\Xi \vdash \widehat{\sigma} : UC_1$ and $\Xi \vdash \widehat{\sigma} : UC_2$,*

1. *$\Xi \vdash UC_1$ & $UC_2 = UC$ is defined and*
2. *$\Xi \vdash \widehat{\sigma} : UC$.*

Proof The proof repeats the proof of Lemma 91 for cases uses Lemma 70 instead of Lemma 90. ■

C.3.6 Unification

Observation 7 (Unification Determinism)**.**

+ *If* $\Theta; \Xi \vDash P \overset{u}{\simeq} Q \dashv UC$ *and* $\Theta; \Xi \vDash P \overset{u}{\simeq} Q \dashv UC'$ *then* $UC = UC'$.
− *If* $\Theta; \Xi \vDash N \overset{u}{\simeq} M \dashv UC$ *and* $\Theta; \Xi \vDash N \overset{u}{\simeq} M \dashv UC'$ *then* $UC = UC'$.

Proof We prove it by mutual structural induction on $\Theta; \Xi \vDash P \overset{u}{\simeq} Q \dashv UC$ and $\Theta; \Xi \vDash N \overset{u}{\simeq} M \dashv UC'$. Let us consider the positive case only since the negative case is symmetric.

First, notice that the rule applied the last is uniquely determined by the shape of P and Q. Second, the premises of each rule are deterministic on the input either by the induction hypothesis or by Observation 6. ■

Lemma 72 (Soundness of Unification)**.**

+ *For normalized* P *and* Q *such that* $\Theta\,;\mathtt{dom}\,(\Xi) \vdash P$ *and* $\Theta \vdash Q$,
if $\Theta; \Xi \vDash P \overset{u}{\simeq} Q \dashv UC$ *then* $\Xi \vdash UC : \mathtt{fav} P$ *and for any normalized* $\widehat{\sigma}$ *such that* $\Xi \vdash \widehat{\sigma} : UC$, $[\widehat{\sigma}]P = Q$.
− *For normalized* N *and* M *such that* $\Theta\,;\mathtt{dom}\,(\Xi) \vdash N$ *and* $\Theta \vdash M$,
if $\Theta; \Xi \vDash N \overset{u}{\simeq} M \dashv UC$ *then* $\Xi \vdash UC : \mathtt{fav} N$ *and for any normalized* $\widehat{\sigma}$ *such that* $\Xi \vdash \widehat{\sigma} : UC$, $[\widehat{\sigma}]N = M$.

Proof We prove by induction on the derivation of $\Theta; \Xi \vDash N \overset{u}{\simeq} M \dashv UC$ and mutually $\Theta; \Xi \vDash P \overset{u}{\simeq} Q \dashv UC$. Let us consider the last rule forming this derivation.

Case 1. ($\textsc{Var}^{\overset{u}{\simeq}}_{-}$), then $N = \alpha^- = M$. The resulting unification constraint is empty: $UC = \cdot$. It satisfies $\Xi \vdash UC : \cdot$ vacuously, and $[\widehat{\sigma}]\alpha^- = \alpha^-$, that is $[\widehat{\sigma}]N = M$.

Case 2. ($\uparrow^{\overset{u}{\simeq}}$), then $N = \uparrow P$ and $M = \uparrow Q$. The algorithm makes a recursive call to $\Theta; \Xi \vDash P \overset{u}{\simeq} Q \dashv UC$ returning UC. By induction hypothesis, $\Xi \vdash UC : \mathtt{fav} P$ and thus, $\Xi \vdash UC : \mathtt{fav}{\uparrow}P$, and for any $\Xi \vdash \widehat{\sigma} : UC$, $[\widehat{\sigma}]N = [\widehat{\sigma}]{\uparrow}P = {\uparrow}[\widehat{\sigma}]P = {\uparrow}Q = M$, as required.

Case 3. ($\to^{\overset{u}{\simeq}}$), then $N = P \to N'$ and $M = Q \to M'$. The algorithm makes two recursive calls to $\Theta; \Xi \vDash P \overset{u}{\simeq} Q \dashv UC_1$ and $\Theta; \Xi \vDash N' \overset{u}{\simeq} M' \dashv UC_2$ returning $\Xi \vdash UC_1 \,\&\, UC_2 = UC$ as the result.

It is clear that P, N', Q, and M' are normalized, and that $\Theta\,;\mathtt{dom}\,(\Xi) \vdash P$, $\Theta\,;\mathtt{dom}\,(\Xi) \vdash N'$, $\Theta \vdash Q$, and $\Theta \vdash M'$. This way, the induction hypothesis is applicable to both recursive calls.

By applying the induction hypothesis to $\Theta; \Xi \vDash P \overset{u}{\simeq} Q \dashv UC_1$, we have:

- $\Xi \vdash UC_1 : \mathtt{fav} P$,
- for any $\Xi \vdash \widehat{\sigma}' : UC_1$, $[\widehat{\sigma}']P = Q$.

By applying it to $\Theta; \Xi \vdash N' \overset{u}{\simeq} M' \dashv UC_2$, we have:

- $\Xi \vdash UC_2 : \mathtt{fav} N'$,
- for any $\Xi \vdash \widehat{\sigma}' : UC_2$, $[\widehat{\sigma}']N' = M'$.

Let us take an arbitrary $\Xi \vdash \widehat{\sigma} : UC$. By the soundness of the constraint merge (Lemma 89), $\Xi \vdash UC_1 \,\&\, UC_2 = UC$ implies $\Xi \vdash \widehat{\sigma} : UC_1$ and $\Xi \vdash \widehat{\sigma} : UC_2$.

Applying the induction hypothesis to $\Xi \vdash \widehat{\sigma} : UC_1$, we have $[\widehat{\sigma}]P = Q$; applying it to $\Xi \vdash \widehat{\sigma} : UC_2$, we have $[\widehat{\sigma}]N' = M'$. This way, $[\widehat{\sigma}]N = [\widehat{\sigma}]P \to [\widehat{\sigma}]N' = Q \to M' = M$.

Case 4. $(\forall^{\overset{u}{\simeq}})$, then $N = \forall\overrightarrow{\alpha^+}.\, N'$ and $M = \forall\overrightarrow{\alpha^+}.\, M'$. The algorithm makes a recursive call to $\Theta, \overrightarrow{\alpha^+}; \Xi \vdash N' \overset{u}{\simeq} M' \dashv UC$ returning UC as the result.

The induction hypothesis is applicable: $\Theta, \overrightarrow{\alpha^+}; \mathtt{dom}(\Xi) \vdash N'$ and $\Theta, \overrightarrow{\alpha^+} \vdash M'$ hold by inversion, and N' and M' are normalized, since N and M are. Let us take an arbitrary $\Xi \vdash \widehat{\sigma} : UC$. By the induction hypothesis, $[\widehat{\sigma}]N' = M'$. Then $[\widehat{\sigma}]N = [\widehat{\sigma}]\forall\overrightarrow{\alpha^+}.\, N' = \forall\overrightarrow{\alpha^+}.\, [\widehat{\sigma}]N' = \forall\overrightarrow{\alpha^+}.\, M' = M$.

Case 5. $(\mathrm{UVAR}^{\overset{u}{\simeq}}_{-})$, then $N = \widehat{\alpha}^-$, $\widehat{\alpha}^-\{\Theta_0\} \in \Xi$, and $\Theta_0 \vdash M$. As the result, the algorithm returns $UC = (\widehat{\alpha}^- :\simeq M)$.

It is clear that $\widehat{\alpha}^-\{\Theta_0\} \vdash (\widehat{\alpha}^- :\simeq M)$, since $\Theta_0 \vdash M$, meaning that $\Xi \vdash UC$.

Let us take an arbitrary $\widehat{\sigma}$ such that $\Xi \vdash \widehat{\sigma} : UC$. Since $UC = (\widehat{\alpha}^- :\simeq M)$, $\Xi \vdash \widehat{\sigma} : UC$ implies $\Xi(\widehat{\alpha}^-) \vdash [\widehat{\sigma}]\widehat{\alpha}^- : (\widehat{\alpha}^- :\simeq M)$. By inversion of $(:\simeq^{\mathrm{SAT}}_{-})$, it means $\Xi(\widehat{\alpha}^-) \vdash [\widehat{\sigma}]\widehat{\alpha}^- \simeq^{\leq} M$. This way, $\Xi(\widehat{\alpha}^-) \vdash [\widehat{\sigma}]N \simeq^{\leq} M$. Notice that $\widehat{\sigma}$ and N are normalized, and by Lemma 43, so is $[\widehat{\sigma}]N$. Since both sides of $\Xi(\widehat{\alpha}^-) \vdash [\widehat{\sigma}]N \simeq^{\leq} M$ are normalized, by Lemma 48, we have $[\widehat{\sigma}]N = M$.

Case 6. The positive cases are proved symmetrically.

■

Lemma 73 (Completeness of Unification)**.**

+ *For normalized P and Q such that $\Theta\,;\mathtt{dom}(\Xi) \vdash P$ and $\Theta \vdash Q$, suppose that there exists $\Xi \vdash \widehat{\sigma} : \mathtt{fav}(P)$ such that $[\widehat{\sigma}]P = Q$, then $\Theta; \Xi \vdash P \overset{u}{\simeq} Q \dashv UC$ for some UC.*
− *For normalized N and M such that $\Theta\,;\mathtt{dom}(\Xi) \vdash N$ and $\Theta \vdash M$, suppose that there exists $\Xi \vdash \widehat{\sigma} : \mathtt{fav}(N)$ such that $[\widehat{\sigma}]N = M$, then $\Theta; \Xi \vdash N \overset{u}{\simeq} M \dashv UC$ for some UC.*

Proof We prove it by induction on the structure of P and mutually, N.

Case 1. $N = \widehat{\alpha}^-$

$\Theta\,;\mathtt{dom}(\Xi) \vdash \widehat{\alpha}^-$ means that $\widehat{\alpha}^-\{\Theta_0\} \in \Xi$ for some Θ_0.

Let us take an arbitrary $\Xi \vdash \widehat{\sigma} : \widehat{\alpha}^-$ such that $[\widehat{\sigma}]\widehat{\alpha}^- = M$. $\Xi \vdash \widehat{\sigma} : \widehat{\alpha}^-$ means that $\Theta_0 \vdash M$. This way, $(\mathrm{UVAR}^{\overset{u}{\simeq}}_{-})$ is applicable to infer $\Theta; \Xi \vdash \widehat{\alpha}^- \overset{u}{\simeq} M \dashv (\widehat{\alpha}^- :\simeq M)$.

Case 2. $N = \alpha^-$

Let us take an arbitrary $\Xi \vdash \widehat{\sigma} : \mathtt{fav}(\alpha^-)$ such that $[\widehat{\sigma}]\alpha^- = M$. Since $\mathtt{fav}(\alpha^-) = \emptyset$, $[\widehat{\sigma}]\alpha^- = M$ means $M = \alpha^-$.

This way, $(\textsc{Var}^{\overset{u}{\simeq}}_-)$ infers $\Theta; \Xi \vDash \alpha^- \overset{u}{\simeq} \alpha^- \dashv \cdot$, which is rewritten as $\Theta; \Xi \vDash N \overset{u}{\simeq} M \dashv \cdot$.

Case 3. $N = \uparrow P$

Let us take an arbitrary $\Xi \vdash \widehat{\sigma} : \mathtt{fav}(P)$ such that $[\widehat{\sigma}]\uparrow P = M$. The latter means $\uparrow[\widehat{\sigma}]P = M$, i.e. $M = \uparrow Q$ for some Q and $[\widehat{\sigma}]P = Q$.

Let us show that the induction hypothesis is applicable to $[\widehat{\sigma}]P = Q$. Notice that P is normalized, since $N = \uparrow P$ is normalized, $\Theta\,;\mathtt{dom}\,(\Xi) \vdash P$ holds by inversion of $\Theta\,;\mathtt{dom}\,(\Xi) \vdash \uparrow P$, and $\Theta \vdash Q$ holds by inversion of $\Theta \vdash \uparrow Q$.

This way, by the induction hypothesis there exists UC such that $\Theta; \Xi \vDash P \overset{u}{\simeq} Q \dashv UC$.

Case 4. $N = P \to N'$

Let us take an arbitrary $\Xi \vdash \widehat{\sigma} : \mathtt{fav}(P \to N')$ such that $[\widehat{\sigma}](P \to N') = M$. The latter means $[\widehat{\sigma}]P \to [\widehat{\sigma}]N' = M$, i.e. $M = Q \to M'$ for some Q and M', such that $[\widehat{\sigma}]P = Q$ and $[\widehat{\sigma}]N' = M'$.

Let us show that the induction hypothesis is applicable to $\Xi \vdash \widehat{\sigma}|_{\mathtt{fav}(P)} : \mathtt{fav}(P)$ and $[\widehat{\sigma}|_{\mathtt{fav}(P)}]P = Q$ (the latter holds since $[\widehat{\sigma}|_{\mathtt{fav}(P)}]P = [\widehat{\sigma}]P$ by Lemma 63),

- P is normalized, since $N = P \to N'$ is normalized
- $\Theta\,;\mathtt{dom}\,(\Xi) \vdash P$ follows from the inversion of $\Theta\,;\mathtt{dom}\,(\Xi) \vdash P \to N'$,
- $\Theta \vdash Q$.

Then by the induction hypothesis, $\Theta; \Xi \vDash P \overset{u}{\simeq} Q \dashv UC_1$. Analogously, the induction hypothesis is applicable to $[\widehat{\sigma}|_{\mathtt{fav}N'}]N' = M'$, and thus, $\Theta; \Xi \vDash N' \overset{u}{\simeq} M' \dashv UC_2$.

To apply $(\to^{\overset{u}{\simeq}})$ and infer the required $\Theta; \Xi \vDash N \overset{u}{\simeq} M \dashv UC$, it is left to show that $\Xi \vdash UC_1 \,\&\, UC_2 = UC$. It holds by completeness of the unification constraint merge (Lemma 71) for $\Xi \vdash UC_1 : \mathtt{fav}P$, $\Xi \vdash UC_2 : \mathtt{fav}N'$ (which hold by soundness), and $\Xi \vdash \widehat{\sigma} : \mathtt{fav}(P) \cup \mathtt{fav}(N')$, which holds since $\mathtt{fav}(P) \cup \mathtt{fav}(N') = \mathtt{fav}(P \to N')$. Notice that by soundness, $\Xi \vdash \widehat{\sigma}|_{\mathtt{fav}(P)} : UC_1$, which implies $\Xi \vdash \widehat{\sigma} : UC_1$. Analogously, $\Xi \vdash \widehat{\sigma} : UC_2$.

Case 5. $N = \forall \overrightarrow{\alpha^+}.\, N'$

Let us take an arbitrary $\Xi \vdash \widehat{\sigma} : \mathtt{fav}(N')$ such that $[\widehat{\sigma}]\forall \overrightarrow{\alpha^+}.\, N' = M$. The latter means $\forall \overrightarrow{\alpha^+}.\, [\widehat{\sigma}]N' = M$, i.e. $M = \forall \overrightarrow{\alpha^+}.\, M'$ for some M' such that $[\widehat{\sigma}]N' = M'$.

Let us show that the induction hypothesis is applicable to $[\widehat{\sigma}]N' = M'$. Notice that N' is normalized, since $N = \forall \overrightarrow{\alpha^+}.\, N'$ is normalized, $\Theta, \overrightarrow{\alpha^+}\,;\mathtt{dom}\,(\Xi) \vdash N'$ follows from inversion of $\Theta\,;\mathtt{dom}\,(\Xi) \vdash \forall \overrightarrow{\alpha^+}.\, N'$, $\Theta, \overrightarrow{\alpha^+} \vdash M'$ follows from inversion of $\Theta \vdash \forall \overrightarrow{\alpha^+}.\, M'$, and $\Xi \vdash \widehat{\sigma} : \mathtt{fav}(N')$ by assumption.

This way, by the induction hypothesis, $\Theta, \overrightarrow{\alpha^+}; \Xi \vDash N' \overset{u}{\simeq} M' \dashv UC$ exists and moreover, $\Xi \vdash \widehat{\sigma} : UC$. Hence, $(\forall^{\overset{u}{\simeq}})$ is applicable to infer $\Theta; \Xi \vDash \forall \overrightarrow{\alpha^+}.\, N' \overset{u}{\simeq} \forall \overrightarrow{\alpha^+}.\, M' \dashv UC$, that is $\Theta; \Xi \vDash N \overset{u}{\simeq} M \dashv UC$.

Case 6. The positive cases are proved symmetrically.

■

C.3.7 Anti-unification

Observation 8 (Determinism of Anti-unification Algorithm)**.**

+ *If* $\Theta \vDash P_1 \overset{a}{\simeq} P_2 \dashv (\widehat{\Theta}, Q, \widehat{\tau}_1, \widehat{\tau}_2)$ *and* $\Theta \vDash P_1 \overset{a}{\simeq} P_2 \dashv (\widehat{\Theta}', Q', \widehat{\tau}_1', \widehat{\tau}_2')$*, then* $\widehat{\Theta} = \widehat{\Theta}'$*,* $Q = Q'$*,* $\widehat{\tau}_1 = \widehat{\tau}_1'$*, and* $\widehat{\tau}_2 = \widehat{\tau}_2'$*.*
– *If* $\Theta \vDash N_1 \overset{a}{\simeq} N_2 \dashv (\widehat{\Theta}, M, \widehat{\tau}_1, \widehat{\tau}_2)$ *and* $\Theta \vDash N_1 \overset{a}{\simeq} N_2 \dashv (\widehat{\Theta}', M', \widehat{\tau}_1', \widehat{\tau}_2')$*, then* $\widehat{\Theta} = \widehat{\Theta}'$*,* $M = M'$*,* $\widehat{\tau}_1 = \widehat{\tau}_1'$*, and* $\widehat{\tau}_2 = \widehat{\tau}_2'$*.*

Proof By trivial induction on $\Theta \vDash P_1 \overset{a}{\simeq} P_2 \dashv (\widehat{\Theta}, Q, \widehat{\tau}_1, \widehat{\tau}_2)$ and mutually on $\Theta \vDash N_1 \overset{a}{\simeq} N_2 \dashv (\widehat{\Theta}, M, \widehat{\tau}_1, \widehat{\tau}_2)$. ■

Observation 9 (Uniqueness of Anti-unification Variable Names)**.** *Names of the anti-unification variables are uniquely defined by the types they are mapped to by the resulting substitutions.*

+ *Assuming* P_1 *and* P_2 *are normalized, if* $\Theta \vDash P_1 \overset{a}{\simeq} P_2 \dashv (\widehat{\Theta}, Q, \widehat{\tau}_1, \widehat{\tau}_2)$ *then for any* $\widehat{\beta}^- \in \widehat{\Theta}$*,* $\widehat{\beta}^- = \widehat{\alpha}^-_{\{[\widehat{\tau}_1]\widehat{\beta}^-, [\widehat{\tau}_2]\widehat{\beta}^-\}}$
– *Assuming* N_1 *and* N_2 *are normalized, if* $\Theta \vDash N_1 \overset{a}{\simeq} N_2 \dashv (\widehat{\Theta}, M, \widehat{\tau}_1, \widehat{\tau}_2)$ *then for any* $\widehat{\beta}^- \in \widehat{\Theta}$*,* $\widehat{\beta}^- = \widehat{\alpha}^-_{\{[\widehat{\tau}_1]\widehat{\beta}^-, [\widehat{\tau}_2]\widehat{\beta}^-\}}$

Proof By simple induction on $\Theta \vDash P_1 \overset{a}{\simeq} P_2 \dashv (\widehat{\Theta}, Q, \widehat{\tau}_1, \widehat{\tau}_2)$ and mutually on $\Theta \vDash N_1 \overset{a}{\simeq} N_2 \dashv (\widehat{\Theta}, M, \widehat{\tau}_1, \widehat{\tau}_2)$. Let us consider the last rule applied to infer this judgment.

Case 1. ($\textsc{Var}^{\overset{a}{\simeq}}_+$) or ($\textsc{Var}^{\overset{a}{\simeq}}_-$), then $\widehat{\Theta} = \cdot$, and the property holds vacuously.

Case 2. (AU) Then $\widehat{\Theta} = \widehat{\alpha}^-_{\{N_1, N_2\}}$, $\widehat{\tau}_1 = \widehat{\alpha}^-_{\{N_1, N_2\}} \mapsto N_1$, and $\widehat{\tau}_2 = \widehat{\alpha}^-_{\{N_1, N_2\}} \mapsto N_2$. So the property holds trivially.

Case 3. ($\rightarrow^{\overset{a}{\simeq}}$) In this case, $\widehat{\Theta} = \widehat{\Theta}' \cup \widehat{\Theta}''$, $\widehat{\tau}_1 = \widehat{\tau}_1' \cup \widehat{\tau}_1''$, and $\widehat{\tau}_2 = \widehat{\tau}_2' \cup \widehat{\tau}_2''$, where the property holds for $(\widehat{\Theta}', \widehat{\tau}_1', \widehat{\tau}_2')$ and $(\widehat{\Theta}'', \widehat{\tau}_1'', \widehat{\tau}_2'')$ by the induction hypothesis. Then since the union of solutions does not change the types the variables are mapped to, the required property holds for $\widehat{\Theta}$, $\widehat{\tau}_1$, and $\widehat{\tau}_2$.

Case 4. For the other rules, the resulting $\widehat{\Theta}$ is taken from the recursive call and the required property holds immediately by the induction hypothesis.

■

Lemma 74 (Soundness of Anti-Unification)**.**

+ *Assuming* P_1 *and* P_2 *are normalized, if* $\Theta \vDash P_1 \overset{a}{\simeq} P_2 \dashv (\widehat{\Theta}, Q, \widehat{\tau}_1, \widehat{\tau}_2)$ *then*

1. $\Theta ; \widehat{\Theta} \vdash Q$,
2. $\Theta ; \cdot \vdash \widehat{\tau}_i : \widehat{\Theta}$ *for* $i \in \{1, 2\}$ *are anti-unification substitutions, and*
3. $[\widehat{\tau}_i] Q = P_i$ *for* $i \in \{1, 2\}$.

– *Assuming* N_1 *and* N_2 *are normalized, if* $\Theta \vDash N_1 \overset{a}{\simeq} N_2 \dashv (\widehat{\Theta}, M, \widehat{\tau}_1, \widehat{\tau}_2)$ *then*

1. $\Theta ; \widehat{\Theta} \vdash M$,
2. $\Theta ; \cdot \vdash \widehat{\tau}_i : \widehat{\Theta}$ *for* $i \in \{1, 2\}$ *are anti-unification substitutions, and*
3. $[\widehat{\tau}_i] M = N_i$ *for* $i \in \{1, 2\}$.

Proof We prove it by induction on $\Theta \vDash N_1 \overset{a}{\simeq} N_2 \dashv (\widehat{\Theta}, M, \widehat{\tau}_1, \widehat{\tau}_2)$ and mutually, $\Theta \vDash P_1 \overset{a}{\simeq} P_2 \dashv (\widehat{\Theta}, Q, \widehat{\tau}_1, \widehat{\tau}_2)$. Let us consider the last rule applied to infer this judgement.

Case 1. ($\textsc{Var}^{\overset{a}{\simeq}}_{-}$), then $N_1 = \alpha^- = N_2$, $\widehat{\Theta} = \cdot$, $M = \alpha^-$, and $\widehat{\tau}_1 = \widehat{\tau}_2 = \cdot$.

1. $\Theta ; \cdot \vdash \alpha^-$ follows from the assumption $\Theta \vdash \alpha^-$,
2. $\Theta ; \cdot \vdash \cdot : \cdot$ holds trivially, and
3. $[\cdot]\alpha^- = \alpha^-$ holds trivially.

Case 2. ($\uparrow^{\overset{a}{\simeq}}$), then $N_1 = \uparrow P_1$, $N_2 = \uparrow P_2$, and the algorithm makes the recursive call: $\Theta \vDash P_1 \overset{a}{\simeq} P_2 \dashv (\widehat{\Theta}, Q, \widehat{\tau}_1, \widehat{\tau}_2)$, returning $(\widehat{\Theta}, \uparrow Q, \widehat{\tau}_1, \widehat{\tau}_2)$ as the result.

Since $N_1 = \uparrow P_1$ and $N_2 = \uparrow P_2$ are normalized, so are P_1 and P_2, and thus, the induction hypothesis is applicable to $\Theta \vDash P_1 \overset{a}{\simeq} P_2 \dashv (\widehat{\Theta}, Q, \widehat{\tau}_1, \widehat{\tau}_2)$:

1. $\Theta ; \widehat{\Theta} \vdash Q$, and hence, $\Theta ; \widehat{\Theta} \vdash \uparrow Q$,
2. $\Theta ; \cdot \vdash \widehat{\tau}_i : \widehat{\Theta}$ for $i \in \{1, 2\}$, and
3. $[\widehat{\tau}_i] Q = P_i$ for $i \in \{1, 2\}$, and then by the definition of the substitution, $[\widehat{\tau}_i] \uparrow Q = \uparrow P_i$ for $i \in \{1, 2\}$.

Case 3. ($\rightarrow^{\overset{a}{\simeq}}$), then $N_1 = P_1 \to N'_1$, $N_2 = P_2 \to N'_2$, and the algorithm makes two recursive calls: $\Theta \vDash P_1 \overset{a}{\simeq} P_2 \dashv (\widehat{\Theta}, Q, \widehat{\tau}_1, \widehat{\tau}_2)$ and $\Theta \vDash N'_1 \overset{a}{\simeq} N'_2 \dashv (\widehat{\Theta}', M, \widehat{\tau}'_1, \widehat{\tau}'_2)$ and and returns $(\widehat{\Theta} \cup \widehat{\Theta}', Q \to M, \widehat{\tau}_1 \cup \widehat{\tau}'_1, \widehat{\tau}_2 \cup \widehat{\tau}'_2)$ as the result.

Notice that the induction hypothesis is applicable to $\Theta \vDash P_1 \overset{a}{\simeq} P_2 \dashv (\widehat{\Theta}, Q, \widehat{\tau}_1, \widehat{\tau}_2)$: P_1 and P_2 are normalized, since $N_1 = P_1 \to N'_1$ and $N_2 = P_2 \to N'_2$ are normalized. Similarly, the induction hypothesis is applicable to $\Theta \vDash N'_1 \overset{a}{\simeq} N'_2 \dashv (\widehat{\Theta}', M, \widehat{\tau}'_1, \widehat{\tau}'_2)$.

This way, by the induction hypothesis:

1. $\Theta ; \widehat{\Theta} \vdash Q$ and $\Theta ; \widehat{\Theta}' \vdash M$. Then by weakening (Corollary 22), $\Theta ; \widehat{\Theta} \cup \widehat{\Theta}' \vdash Q$ and $\Theta ; \widehat{\Theta} \cup \widehat{\Theta}' \vdash M$, which implies $\Theta ; \widehat{\Theta} \cup \widehat{\Theta}' \vdash Q \to M$;
2. $\Theta ; \cdot \vdash \widehat{\tau}_i : \widehat{\Theta}$ and $\Theta ; \cdot \vdash \widehat{\tau}'_i : \widehat{\Theta}'$ Then $\Theta ; \cdot \vdash \widehat{\tau}_i \cup \widehat{\tau}'_i : \widehat{\Theta} \cup \widehat{\Theta}'$ are well-defined anti-unification substitutions. Let us take an arbitrary $\widehat{\beta}^- \in \widehat{\Theta} \cup \widehat{\Theta}'$. If $\widehat{\beta}^- \in \widehat{\Theta}$. then $\Theta ; \cdot \vdash \widehat{\tau}_i : \widehat{\Theta}$ implies that $\widehat{\tau}_i$, and hence, $\widehat{\tau}_i \cup \widehat{\tau}'_i$ contains an entry well-formed in Θ. If $\widehat{\beta}^- \in \widehat{\Theta}'$, the reasoning is symmetric.

$\widehat{\tau}_i \cup \widehat{\tau}'_i$ is a well-defined anti-unification substitution: any anti-unification variable occurs uniquely $\widehat{\tau}_i \cup \widehat{\tau}'_i$, since by Observation 9, the name of the variable is in one-to-one correspondence with the pair of types it is mapped to by $\widehat{\tau}_1$ and $\widehat{\tau}_2$, an is in one-to-one correspondence with the pair of types it is mapped to by $\widehat{\tau}'_1$ and $\widehat{\tau}'_2$ i.e. if $\widehat{\beta}^- \in \widehat{\Theta} \cap \widehat{\Theta}'$ then $[\widehat{\tau}_1]\widehat{\beta}^- = [\widehat{\tau}'_1]\widehat{\beta}^-$, and $[\widehat{\tau}_2]\widehat{\beta}^- = [\widehat{\tau}'_2]\widehat{\beta}^-$.

3. $[\widehat{\tau}_i]Q = P_i$ and $[\widehat{\tau}'_i]M = N'_i$. Since $\widehat{\tau}_i \cup \widehat{\tau}'_i$ restricted to $\widehat{\Theta}$ is $\widehat{\tau}_i$, and $\widehat{\tau}_i \cup \widehat{\tau}'_i$ restricted to $\widehat{\Theta}'$ is $\widehat{\tau}'_i$, we have $[\widehat{\tau}_i \cup \widehat{\tau}'_i]Q = P_i$ and $[\widehat{\tau}_i \cup \widehat{\tau}'_i]M = N'_i$, and thus, $[\widehat{\tau}_i \cup \widehat{\tau}'_i]Q \to M = P_1 \to N'_1$

Case 4. ($\forall^{\overset{a}{\simeq}}$), then $N_1 = \forall\overrightarrow{\alpha^+}.\, N'_1$, $N_2 = \forall\overrightarrow{\alpha^+}.\, N'_2$, and the algorithm makes a recursive call: $\Theta \vDash N'_1 \overset{a}{\simeq} N'_2 \dashv (\widehat{\Theta}, M, \widehat{\tau}_1, \widehat{\tau}_2)$ and returns $(\widehat{\Theta}, \forall\overrightarrow{\alpha^+}.\, M, \widehat{\tau}_1, \widehat{\tau}_2)$ as the result.

Similarly to case 2, we apply the induction hypothesis to $\Theta \vDash N'_1 \overset{a}{\simeq} N'_2 \dashv (\widehat{\Theta}, M, \widehat{\tau}_1, \widehat{\tau}_2)$ to obtain:

1. $\Theta\,;\widehat{\Theta} \vdash M$, and hence, $\Theta\,;\widehat{\Theta} \vdash \forall\overrightarrow{\alpha^+}.\, M$;
2. $\Theta\,;\cdot \vdash \widehat{\tau}_i : \widehat{\Theta}$ for $i \in \{1,2\}$, and
3. $[\widehat{\tau}_i]M = N'_i$ for $i \in \{1,2\}$, and then by the definition of the substitution, $[\widehat{\tau}_i]\forall\overrightarrow{\alpha^+}.\, M = \forall\overrightarrow{\alpha^+}.\, N'_i$ for $i \in \{1,2\}$.

Case 5. (AU), which applies when other rules do not, and $\Theta \vdash N_i$, returning as the result $(\widehat{\Theta}, M, \widehat{\tau}_1, \widehat{\tau}_2) = (\widehat{\alpha}^-_{\{N_1,N_2\}}, \widehat{\alpha}^-_{\{N_1,N_2\}}, (\widehat{\alpha}^-_{\{N_1,N_2\}} \mapsto N_1), (\widehat{\alpha}^-_{\{N_1,N_2\}} \mapsto N_2))$.

1. $\Theta\,;\widehat{\Theta} \vdash M$ is rewritten as $\Theta\,;\widehat{\alpha}^-_{\{N_1,N_2\}} \vdash \widehat{\alpha}^-_{\{N_1,N_2\}}$, which holds trivially;
2. $\Theta\,;\cdot \vdash \widehat{\tau}_i : \widehat{\Theta}$ is rewritten as $\Theta\,;\cdot \vdash (\widehat{\alpha}^-_{\{N_1,N_2\}} \mapsto N_i) : \widehat{\alpha}^-_{\{N_1,N_2\}}$, which holds since $\Theta \vdash N_i$ by the premise of the rule;
3. $[\widehat{\tau}_i]M = N_i$ is rewritten as $[\widehat{\alpha}^-_{\{N_1,N_2\}} \mapsto N_i]\widehat{\alpha}^-_{\{N_1,N_2\}} = N_i$, which holds trivially by the definition of substitution.

Case 6. Positive cases are proved symmetrically.

■

Lemma 75 (Completeness of Anti-Unification)**.**

\+ *Assume that P_1 and P_2 are normalized, and there exists $(\widehat{\Theta}', Q', \widehat{\tau}'_1, \widehat{\tau}'_2)$ such that*

1. $\Theta\,;\widehat{\Theta}' \vdash Q'$,
2. $\Theta\,;\cdot \vdash \widehat{\tau}'_i : \widehat{\Theta}'$ *for* $i \in \{1,2\}$ *are anti-unification substitutions, and*
3. $[\widehat{\tau}'_i]Q' = P_i$ *for* $i \in \{1,2\}$.

Then the anti-unification algorithm terminates, that is there exists $(\widehat{\Theta}, Q, \widehat{\tau}_1, \widehat{\tau}_2)$ such that $\Theta \vDash P_1 \overset{a}{\simeq} P_2 \dashv (\widehat{\Theta}, Q, \widehat{\tau}_1, \widehat{\tau}_2)$

\− *Assume that N_1 and N_2 are normalized, and there exists $(\widehat{\Theta}', M', \widehat{\tau}'_1, \widehat{\tau}'_2)$ such that*

1. $\Theta\,;\widehat{\Theta}' \vdash M'$,

2. $\Theta ; \cdot \vdash \widehat{\tau}'_i : \widehat{\Theta}'$ *for* $i \in \{1, 2\}$, *are anti-unification substitutions, and*

3. $[\widehat{\tau}'_i]M' = N_i$ *for* $i \in \{1, 2\}$.

Then the anti-unification algorithm succeeds, that is there exists $(\widehat{\Theta}, M, \widehat{\tau}_1, \widehat{\tau}_2)$ *such that* $\Theta \vDash N_1 \overset{a}{\simeq} N_2 \dashv (\widehat{\Theta}, M, \widehat{\tau}_1, \widehat{\tau}_2)$.

Proof We prove it by the induction on M' and mutually on Q'.

Case 1. $M' = \widehat{\alpha}^-$ Then since $\Theta ; \cdot \vdash \widehat{\tau}'_i : \widehat{\Theta}'$, $\Theta \vdash [\widehat{\tau}'_i]M' = N_i$. This way, (AU) is always applicable if other rules are not.

Case 2. $M' = \alpha^-$ Then $\alpha^- = [\widehat{\tau}'_i]\alpha^- = N_i$, which means that $(\text{Var}^{a}_{\simeq^-})$ is applicable.

Case 3. $M' = \uparrow Q'$ Then $\uparrow[\widehat{\tau}'_i]Q' = [\widehat{\tau}'_i]\uparrow Q' = N_i$, that is N_1 and N_2 have form $\uparrow P_1$ and $\uparrow P_2$ respectively.

Moreover, $[\widehat{\tau}'_i]Q' = P_i$, which means that $(\widehat{\Theta}', Q', \widehat{\tau}'_1, \widehat{\tau}'_2)$ is an anti-unifier of P_1 and P_2. Then by the induction hypothesis, there exists $(\widehat{\Theta}, Q, \widehat{\tau}_1, \widehat{\tau}_2)$ such that $\Theta \vDash P_1 \overset{a}{\simeq} P_2 \dashv (\widehat{\Theta}, Q, \widehat{\tau}_1, \widehat{\tau}_2)$, and hence, $\Theta \vDash \uparrow P_1 \overset{a}{\simeq} \uparrow P_2 \dashv (\widehat{\Theta}, \uparrow Q, \widehat{\tau}_1, \widehat{\tau}_2)$ by $(\uparrow^{a}_{\simeq})$.

Case 4. $M' = \forall \overrightarrow{\alpha^{\mp}}. M''$ This case is similar to the previous one: we consider $\forall \overrightarrow{\alpha^{\mp}}$ as a constructor. Notice that $\forall \overrightarrow{\alpha^{\mp}}. [\widehat{\tau}'_i]M'' = [\widehat{\tau}'_i]\forall \overrightarrow{\alpha^{\mp}}. M'' = N_i$, that is N_1 and N_2 have form $\forall \overrightarrow{\alpha^{\mp}}. N''_1$ and $\forall \overrightarrow{\alpha^{\mp}}. N''_2$ respectively.

Moreover, $[\widehat{\tau}'_i]M'' = N''_i$, which means that $(\widehat{\Theta}', M'', \widehat{\tau}'_1, \widehat{\tau}'_2)$ is an anti-unifier of N''_1 and N''_2. Then by the induction hypothesis, there exists $(\widehat{\Theta}, M, \widehat{\tau}_1, \widehat{\tau}_2)$ such that $\Theta \vDash N''_1 \overset{a}{\simeq} N''_2 \dashv (\widehat{\Theta}, M, \widehat{\tau}_1, \widehat{\tau}_2)$, and hence, $\Theta \vDash \forall \overrightarrow{\alpha^{\mp}}. N''_1 \overset{a}{\simeq} \forall \overrightarrow{\alpha^{\mp}}. N''_2 \dashv (\widehat{\Theta}, \forall \overrightarrow{\alpha^{\mp}}. M, \widehat{\tau}_1, \widehat{\tau}_2)$ by $(\forall^{a}_{\simeq})$.

Case 5. $M' = Q' \to M''$ Then $[\widehat{\tau}'_i]Q' \to [\widehat{\tau}'_i]M'' = [\widehat{\tau}'_i](Q' \to M'') = N_i$, that is N_1 and N_2 have form $P_1 \to N'_1$ and $P_2 \to N'_2$ respectively.

Moreover, $[\widehat{\tau}'_i]Q' = P_i$ and $[\widehat{\tau}'_i]M'' = N''_i$, which means that $(\widehat{\Theta}', Q', \widehat{\tau}'_1, \widehat{\tau}'_2)$ is an anti-unifier of P_1 and P_2, and $(\widehat{\Theta}', M'', \widehat{\tau}'_1, \widehat{\tau}'_2)$ is an anti-unifier of N''_1 and N''_2. Then by the induction hypothesis, $\Theta \vDash P_1 \overset{a}{\simeq} P_2 \dashv (\widehat{\Theta}_1, Q, \widehat{\tau}_1, \widehat{\tau}_2)$ and $\Theta \vDash N''_1 \overset{a}{\simeq} N''_2 \dashv (\widehat{\Theta}_2, M, \widehat{\tau}_3, \widehat{\tau}_4)$ succeed. The result of the algorithm is $(\widehat{\Theta}_1 \cup \widehat{\Theta}_2, Q \to M, \widehat{\tau}_1 \cup \widehat{\tau}_3, \widehat{\tau}_2 \cup \widehat{\tau}_4)$.

Case 6. $Q' = \widehat{\alpha}^+$ This case if not possible, since $\Theta ; \widehat{\Theta}' \vdash Q'$ means $\widehat{\alpha}^+ \in \widehat{\Theta}'$, but $\widehat{\Theta}'$ can only contain negative variables.

Case 7. Other positive cases are proved symmetrically to the corresponding negative ones.

■

Lemma 76 (Initiality of Anti-Unification)**.**

+ *Assume that* P_1 *and* P_2 *are normalized, and* $\Theta \vDash P_1 \overset{a}{\simeq} P_2 \dashv (\widehat{\Theta}, Q, \widehat{\tau}_1, \widehat{\tau}_2)$, *then* $(\widehat{\Theta}, Q, \widehat{\tau}_1, \widehat{\tau}_2)$ *is more specific than any other sound anti-unifier* $(\widehat{\Theta}', Q', \widehat{\tau}'_1, \widehat{\tau}'_2)$, *i.e. if*

1. $\Theta ; \widehat{\Theta}' \vdash Q'$,
2. $\Theta ; \cdot \vdash \widehat{\tau}_i' : \widehat{\Theta}'$ *for* $i \in \{1, 2\}$, *and*
3. $[\widehat{\tau}_i']\, Q' = P_i$ *for* $i \in \{1, 2\}$

then there exists $\widehat{\rho}$ *such that* $\Theta ; \widehat{\Theta} \vdash \widehat{\rho} : (\widehat{\Theta}'|_{\mathtt{fav} Q'})$ *and* $[\widehat{\rho}]\, Q' = Q$. *Moreover,* $[\widehat{\rho}]\widehat{\beta}^-$ *can be uniquely determined by* $[\widehat{\tau}_1']\widehat{\beta}^-$, $[\widehat{\tau}_2']\widehat{\beta}^-$, *and* Θ.

– *Assume that* N_1 *and* N_2 *are normalized, and* $\Theta \vDash N_1 \overset{a}{\simeq} N_2 \dashv (\widehat{\Theta}, M, \widehat{\tau}_1, \widehat{\tau}_2)$, *then* $(\widehat{\Theta}, M, \widehat{\tau}_1, \widehat{\tau}_2)$ *is more specific than any other sound anti-unifier* $(\widehat{\Theta}', M', \widehat{\tau}_1', \widehat{\tau}_2')$, *i.e. if*

1. $\Theta ; \widehat{\Theta}' \vdash M'$,
2. $\Theta ; \cdot \vdash \widehat{\tau}_i' : \widehat{\Theta}'$ *for* $i \in \{1, 2\}$, *and*
3. $[\widehat{\tau}_i']\, M' = N_i$ *for* $i \in \{1, 2\}$

then there exists $\widehat{\rho}$ *such that* $\Theta ; \widehat{\Theta} \vdash \widehat{\rho} : (\widehat{\Theta}'|_{\mathtt{fav} M'})$ *and* $[\widehat{\rho}]\, M' = M$. *Moreover,* $[\widehat{\rho}]\widehat{\beta}^-$ *can be uniquely determined by* $[\widehat{\tau}_1']\widehat{\beta}^-$, $[\widehat{\tau}_2']\widehat{\beta}^-$, *and* Θ.

Proof First, let us assume that M' is a algorithmic variable $\widehat{\alpha}^-$. Then we can take $\widehat{\rho} = \widehat{\alpha}^- \mapsto M$, which satisfies the required properties:

- $\Theta ; \widehat{\Theta} \vdash \widehat{\rho} : (\widehat{\Theta}'|_{\mathtt{fav} M'})$ holds since $\widehat{\Theta}'|_{\mathtt{fav} M'} = \widehat{\alpha}^-$ and $\Theta ; \widehat{\Theta} \vdash M$ by the soundness of anti-unification (Lemma 74);
- $[\widehat{\rho}]\, M' = M$ holds by construction
- $[\widehat{\rho}]\widehat{\alpha}^- = M$ is the anti-unifier of $N_1 = [\widehat{\tau}_1']\widehat{\alpha}^-$ and $N_2 = [\widehat{\tau}_2']\widehat{\alpha}^-$ in context Θ, and hence, it is uniquely determined by them (Observation 8).

Now, we can assume that M' is not an algorithmic variable. We prove by induction on the derivation of $\Theta \vDash P_1 \overset{a}{\simeq} P_2 \dashv (\widehat{\Theta}, Q, \widehat{\tau}_1, \widehat{\tau}_2)$ and mutually on the derivation of $\Theta \vDash N_1 \overset{a}{\simeq} N_2 \dashv (\widehat{\Theta}, M, \widehat{\tau}_1, \widehat{\tau}_2)$.

Since M' is not a algorithmic variable, the substitution acting on M' preserves its outer constructor. In other words, $[\widehat{\tau}_i']\, M' = N_i$ means that M', N_1 and N_2 have the same outer constructor. Let us consider the algorithmic anti-unification rule corresponding to this constructor, and show that it was successfully applied to anti-unify N_1 and N_2 (or P_1 and P_2).

Case 1. ($\textsc{Var}^{\overset{a}{\simeq}}_{-}$), i.e. $N_1 = \alpha^- = N_2$. This rule is applicable since it has no premises.

Then $\widehat{\Theta} = \cdot$, $M = \alpha^-$, and $\widehat{\tau}_1 = \widehat{\tau}_2 = \cdot$. Since $[\widehat{\tau}_i']\, M' = N_i = \alpha^-$ and M' is not a algorithmic variable, $M' = \alpha^-$. Then we can take $\widehat{\rho} = \cdot$, which satisfies the required properties:

- $\Theta ; \widehat{\Theta} \vdash \widehat{\rho} : (\widehat{\Theta}'|_{\mathtt{fav} M'})$ holds vacuously since $\widehat{\Theta}'|_{\mathtt{fav} M'} = \cdot$;
- $[\widehat{\rho}]\, M' = M$, that is $[\cdot]\alpha^- = \alpha^-$ holds by substitution properties;
- the unique determination of $[\widehat{\rho}]\widehat{\alpha}^-$ for $\widehat{\alpha}^- \in \widehat{\Theta}'|_{\mathtt{fav} M'} = \cdot$ holds vacuously.

Case 2. $(\uparrow^{\overset{a}{\simeq}})$, i.e. $N_1 = \uparrow P_1$ and $N_2 = \uparrow P_2$.

Then since $[\widehat{\tau}_i']\,M' = N_i = \uparrow P_i$ and M' is not a algorithmic variable, $M' = \uparrow Q'$, where $[\widehat{\tau}_i']\,Q' = P_i$. Let us show that $(\widehat{\Theta}', Q', \widehat{\tau}_1', \widehat{\tau}_2')$ is an anti-unifier of P_1 and P_2.

1. $\Theta\,;\widehat{\Theta}' \vdash Q'$ holds by inversion of $\Theta\,;\widehat{\Theta}' \vdash \uparrow Q'$;
2. $\Theta\,;\cdot \vdash \widehat{\tau}_i' : \widehat{\Theta}'$ holds by assumption;
3. $[\widehat{\tau}_i']\,Q' = P_i$ holds by assumption.

This way, by the completeness of anti-unification (Lemma 75), the anti-unification algorithm succeeds on P_1 and P_2: $\Theta \vDash P_1 \overset{a}{\simeq} P_2 \dashv (\widehat{\Theta}, Q, \widehat{\tau}_1, \widehat{\tau}_2)$, which means that $(\uparrow^{\overset{a}{\simeq}})$ is applicable to infer $\Theta \vDash \uparrow P_1 \overset{a}{\simeq} \uparrow P_2 \dashv (\widehat{\Theta}, \uparrow Q, \widehat{\tau}_1, \widehat{\tau}_2)$.

Moreover, by the induction hypothesis, $(\widehat{\Theta}, Q, \widehat{\tau}_1, \widehat{\tau}_2)$ is more specific than $(\widehat{\Theta}', Q', \widehat{\tau}_1', \widehat{\tau}_2')$, which immediately implies that $(\widehat{\Theta}, \uparrow Q, \widehat{\tau}_1, \widehat{\tau}_2)$ is more specific than $(\widehat{\Theta}', \uparrow Q', \widehat{\tau}_1', \widehat{\tau}_2')$ (we keep the same $\widehat{\rho}$).

Case 3. $(\forall^{\overset{a}{\simeq}})$, i.e. $N_1 = \forall \overrightarrow{\alpha^+}.\, N_1'$ and $N_2 = \forall \overrightarrow{\alpha^+}.\, N_2'$. The proof is symmetric to the previous case. Notice that the context Θ is not changed in $(\forall^{\overset{a}{\simeq}})$, as it represents the context in which the anti-unification variables must be instantiated, rather than the context forming the types that are being anti-unified.

Case 4. $(\to^{\overset{a}{\simeq}})$, i.e. $N_1 = P_1 \to N_1'$ and $N_2 = P_2 \to N_2'$.

Then since $[\widehat{\tau}_i']\,M' = N_i = P_i \to N_i'$ and M' is not a algorithmic variable, $M' = Q' \to M''$, where $[\widehat{\tau}_i']\,Q' = P_i$ and $[\widehat{\tau}_i']\,M'' = N_i''$.

Let us show that $(\widehat{\Theta}', Q', \widehat{\tau}_1', \widehat{\tau}_2')$ is an anti-unifier of P_1 and P_2.

1. $\Theta\,;\widehat{\Theta}' \vdash Q'$ holds by inversion of $\Theta\,;\widehat{\Theta}' \vdash Q' \to M''$;
2. $\Theta\,;\cdot \vdash \widehat{\tau}_i' : \widehat{\Theta}'$ holds by assumption;
3. $[\widehat{\tau}_i']\,Q' = P_i$ holds by assumption.

Similarly, $(\widehat{\Theta}', M'', \widehat{\tau}_1', \widehat{\tau}_2')$ is an anti-unifier of N_1'' and N_2''.

Then by the completeness of anti-unification (Lemma 75), the anti-unification algorithm succeeds on P_1 and P_2: $\Theta \vDash P_1 \overset{a}{\simeq} P_2 \dashv (\widehat{\Theta}_1, Q, \widehat{\tau}_1, \widehat{\tau}_2)$; and on N_1' and N_2': $\Theta \vDash N_1'' \overset{a}{\simeq} N_2'' \dashv (\widehat{\Theta}_2, M''', \widehat{\tau}_3, \widehat{\tau}_4)$. Notice that $\widehat{\tau}_1$ & $\widehat{\tau}_3$ and $\widehat{\tau}_2$ & $\widehat{\tau}_4$ are defined, in other words, for any $\widehat{\beta}^{-} \in \widehat{\Theta}_1 \cap \widehat{\Theta}_2$, $[\widehat{\tau}_1]\widehat{\beta}^{-} = [\widehat{\tau}_2]\widehat{\beta}^{-}$ and $[\widehat{\tau}_3]\widehat{\beta}^{-} = [\widehat{\tau}_4]\widehat{\beta}^{-}$, which follows immediately from Observation 9. This way, the algorithm proceeds by applying $(\to^{\overset{a}{\simeq}})$ and returns $(\widehat{\Theta}_1 \cup \widehat{\Theta}_2, Q \to M''', \widehat{\tau}_1 \cup \widehat{\tau}_3, \widehat{\tau}_2 \cup \widehat{\tau}_4)$.

It is left to construct $\widehat{\rho}$ such that $\Theta\,;\widehat{\Theta} \vdash \widehat{\rho} : (\widehat{\Theta}'|_{\mathtt{fav} M'})$ and $[\widehat{\rho}]\,M' = M$. By the induction hypothesis, there exist $\widehat{\rho}_1$ and $\widehat{\rho}_2$ such that $\Theta\,;\widehat{\Theta}_1 \vdash \widehat{\rho}_1 : (\widehat{\Theta}'|_{\mathtt{fav} Q'})$, $\Theta\,;\widehat{\Theta}_2 \vdash \widehat{\rho}_2 : (\widehat{\Theta}'|_{\mathtt{fav} M''})$, $[\widehat{\rho}_1]\,Q' = Q$, and $[\widehat{\rho}_2]\,M'' = M'''$.

Let us show that $\widehat{\rho} = \widehat{\rho}_1 \cup \widehat{\rho}_2$ satisfies the required properties:

- $\Theta ; \widehat{\Theta}_1 \cup \widehat{\Theta}_2 \vdash \widehat{\rho}_1 \cup \widehat{\rho}_2 : (\widehat{\Theta'}|_{\mathtt{fav} M'})$ holds since $\widehat{\Theta'}|_{\mathtt{fav} M'} = \widehat{\Theta'}|_{\mathtt{fav} Q' \to M''} = (\widehat{\Theta'}|_{\mathtt{fav} Q'}) \cup (\widehat{\Theta'}|_{\mathtt{fav} M''})$, $\Theta ; \widehat{\Theta}_1 \vdash \widehat{\rho}_1 : (\widehat{\Theta'}|_{\mathtt{fav} Q'})$ and $\Theta ; \widehat{\Theta}_2 \vdash \widehat{\rho}_2 : (\widehat{\Theta'}|_{\mathtt{fav} M''})$;
- $[\widehat{\rho}] M' = [\widehat{\rho}](Q' \to M'') = [\widehat{\rho}|_{\mathtt{fav} Q'}] Q' \to [\widehat{\rho}|_{\mathtt{fav} M''}] M'' = [\widehat{\rho}_1] Q' \to [\widehat{\rho}_2] M'' = Q \to M''' = M$;
- Since $[\widehat{\rho}]\widehat{\beta^-}$ is either equal to $[\widehat{\rho}_1]\widehat{\beta^-}$ or $[\widehat{\rho}_2]\widehat{\beta^-}$, it inherits their property that it is uniquely determined by $[\widehat{\tau}'_1]\widehat{\beta^-}$, $[\widehat{\tau}'_2]\widehat{\beta^-}$, and Θ.

Case 5. $P_1 = P_2 = \alpha^+$. This case is symmetric to case 1.

Case 6. $P_1 = \downarrow N_1$ and $P_2 = \downarrow N_2$. This case is symmetric to case 2

Case 7. $P_1 = \exists \overrightarrow{\alpha^-}. P'_1$ and $P_2 = \exists \overrightarrow{\alpha^-}. P'_2$. This case is symmetric to case 3

■

C.3.8 Upper Bounds

Observation 10 (Determinism of Least Upper Bound algorithm)**.** *For types* $\Theta \vdash P_1$, *and* $\Theta \vdash P_2$, *if* $\Theta \vDash P_1 \vee P_2 = Q$ *and* $\Theta \vDash P_1 \vee P_2 = Q'$ *then* $Q = Q'$.

Proof The shape of P_1 and P_2 uniquely determines the rule applied to infer the upper bound. By looking at the inference rules, it is easy to see that the result of the least upper bound algorithm depends on

- the inputs of the algorithm (that is P_1, P_2, and Θ), which are fixed;
- the result of the anti-unification algorithm applied to normalized input, which is deterministic by Observation 8;
- the result of the recursive call, which is deterministic by the induction hypothesis.

■

Lemma 77 (Characterization of the Supertypes)**.** *Let us define the set of upper bounds of a positive type* $\mathsf{UB}(P)$ *in the following way:*

$\Theta \vdash P$	$\mathsf{UB}(\Theta \vdash P)$
$\Theta \vdash \beta^+$	$\{\exists \overrightarrow{\alpha^-}. \beta^+ \mid \text{for } \overrightarrow{\alpha^-}\}$
$\Theta \vdash \exists \overrightarrow{\beta^-}. Q$	$\mathsf{UB}(\Theta, \overrightarrow{\beta^-} \vdash Q)$ *not using* $\overrightarrow{\beta^-}$
$\Theta \vdash \downarrow M$	$\left\{ \exists \overrightarrow{\alpha^-}. \downarrow M' \;\middle|\; \begin{array}{l} \text{for } \overrightarrow{\alpha^-}, M', \text{ and } \overrightarrow{N} \text{ s.t.} \\ \Theta \vdash N_i, \Theta, \overrightarrow{\alpha^-} \vdash M', \text{ and } [\overrightarrow{N}/\overrightarrow{\alpha^-}]\downarrow M' \simeq^D \downarrow M \end{array} \right\}$

Then $\mathsf{UB}(\Theta \vdash P) \equiv \{Q \mid \Theta \vdash Q \geqslant P\}$.

Proof By induction on $\Theta \vdash P$.

Case 1. $P = \beta^+$

Immediately from Lemma 19

Case 2. $P = \exists\overrightarrow{\beta^-}.\ P'$

Then if $\Theta \vdash Q \geqslant \exists\overrightarrow{\beta^-}.\ P'$, then by Lemma 18, $\Theta, \overrightarrow{\beta^-} \vdash Q \geqslant P'$, and $\mathtt{fv}\ Q \cap \overrightarrow{\beta^-} = \emptyset$ by the convention. The other direction holds by $(\exists^{\geqslant})$. This way, $\{Q \mid \Theta \vdash Q \geqslant \exists\overrightarrow{\beta^-}.\ P'\} = \{Q \mid \Theta, \overrightarrow{\beta^-} \vdash Q \geqslant P' \text{ s.t. } \mathtt{fv}\ (Q) \cap \overrightarrow{\beta^-} = \emptyset\}$. From the induction hypothesis, the latter is equal to $\mathsf{UB}(\Theta, \overrightarrow{\beta^-} \vdash P')$ not using $\overrightarrow{\beta^-}$, i.e. $\mathsf{UB}(\Theta \vdash \exists\overrightarrow{\beta^-}.\ P')$.

Case 3. $P = \downarrow M$

Then let us consider two subcases upper bounds without outer quantifiers (we denote the corresponding set restriction as $|_{\not\exists}$) and upper bounds with outer quantifiers ($|_{\exists}$). We prove that for both of these groups, the restricted sets are equal.

a. $Q \neq \exists\overrightarrow{\beta^-}.\ Q'$

Then the last applied rule to infer $\Theta \vdash Q \geqslant \downarrow M$ must be $(\downarrow^{\geqslant})$, which means $Q = \downarrow M'$, and by inversion, $\Theta \vdash M' \simeq^{\leqslant} M$, then by Lemma 34 and $(\downarrow^{\simeq^D})$, $\downarrow M' \simeq^D \downarrow M$. This way, $Q = \downarrow M' \in \{\downarrow M' \mid \downarrow M' \simeq^D \downarrow M\} = \mathsf{UB}(\Theta \vdash \downarrow M)|_{\not\exists}$.

In the other direction,

$$\begin{aligned} \downarrow M' \simeq^D \downarrow M &\Rightarrow \Theta \vdash \downarrow M' \simeq^{\leqslant} \downarrow M && \text{by Lemmas 28 and 29} \\ &\Rightarrow \Theta \vdash \downarrow M' \geqslant \downarrow M && \text{by inversion} \end{aligned}$$

b. $Q = \exists\overrightarrow{\beta^-}.\ Q'$ (for non-empty $\overrightarrow{\beta^-}$)

Then the last rule applied to infer $\Theta \vdash \exists\overrightarrow{\beta^-}.\ Q' \geqslant \downarrow M$ must be $(\exists^{\geqslant})$. Inversion of this rule gives us $\Theta \vdash [\overrightarrow{N}/\overrightarrow{\beta^-}]Q' \geqslant \downarrow M$ for some $\Theta \vdash N_i$. Notice that $[\overrightarrow{N}/\overrightarrow{\beta^-}]Q'$ has no outer quantifiers. Thus from case 3.*a*, $[\overrightarrow{N}/\overrightarrow{\beta^-}]Q' \simeq^D \downarrow M$, which is only possible if $Q' = \downarrow M'$. This way, $Q = \exists\overrightarrow{\beta^-}.\ \downarrow M' \in \mathsf{UB}(\Theta \vdash \downarrow M)|_{\exists}$ (notice that $\overrightarrow{\beta^-}$ is not empty).

In the other direction,

$$\begin{aligned} [\overrightarrow{N}/\overrightarrow{\beta^-}]\downarrow M' \simeq^D \downarrow M &\Rightarrow \Theta \vdash [\overrightarrow{N}/\overrightarrow{\beta^-}]\downarrow M' \simeq^{\leqslant} \downarrow M && \text{by Lemmas 28 and 29} \\ &\Rightarrow \Theta \vdash [\overrightarrow{N}/\overrightarrow{\beta^-}]\downarrow M' \geqslant \downarrow M && \text{by inversion} \\ &\Rightarrow \Theta \vdash \exists\overrightarrow{\beta^-}.\ \downarrow M' \geqslant \downarrow M && \text{by } (\exists^{\geqslant}) \end{aligned}$$

■

Lemma 78 (Characterization of the Normalized Supertypes)**.** *For a normalized positive type $P = \mathtt{nf}\ (P)$, let us define the set of normalized upper bounds in the following way:*

$$\frac{}{\Theta \vdash P}\ \mathsf{NFUB}(\Theta \vdash P)$$

$$\frac{}{\Theta \vdash \beta^+}\ \{\beta^+\}$$

$$\frac{}{\Theta \vdash \exists\overrightarrow{\beta^-}.\ P}\ \mathsf{NFUB}(\Theta, \overrightarrow{\beta^-} \vdash P) \text{ not using } \overrightarrow{\beta^-}$$

$$\frac{}{\Theta \vdash \downarrow M}\ \left\{ \exists\overrightarrow{\alpha^-}.\ \downarrow M' \;\middle|\; \begin{array}{l} \text{for } \overrightarrow{\alpha^-},\ M', \text{ and } \overrightarrow{N} \text{ s.t. } \mathtt{ord}\ \overrightarrow{\alpha^-}\ \mathtt{in}\ M' = \overrightarrow{\alpha^-}, \\ \Theta \vdash N_i, \Theta, \overrightarrow{\alpha^-} \vdash M', \text{ and } [\overrightarrow{N}/\overrightarrow{\alpha^-}]\downarrow M' = \downarrow M \end{array} \right\}$$

Then $\mathsf{NFUB}(\Theta \vdash P) \equiv \{\mathtt{nf}\,(Q) \mid \Theta \vdash Q \geqslant P\}$.

Proof By induction on $\Theta \vdash P$.

Case 1. $P = \beta^+$

Then from Lemma 77, $\{\mathtt{nf}\,(Q) \mid \Theta \vdash Q \geqslant \beta^+\} = \{\mathtt{nf}\,(\exists\overrightarrow{\alpha^-}.\,\beta^+) \mid \text{for some } \overrightarrow{\alpha^-}\} = \{\beta^+\}$

Case 2. $P = \exists\overrightarrow{\beta^-}.\,P'$

$\mathsf{NFUB}(\Theta \vdash \exists\overrightarrow{\beta^-}.\,P')$

$= \mathsf{NFUB}(\Theta, \overrightarrow{\beta^-} \vdash P')$ not using $\overrightarrow{\beta^-}$

$= \{\mathtt{nf}\,(Q) \mid \Theta, \overrightarrow{\beta^-} \vdash Q \geqslant P'\}$ not using $\overrightarrow{\beta^-}$ — by the induction hypothesis

$= \{\mathtt{nf}\,(Q) \mid \Theta, \overrightarrow{\beta^-} \vdash Q \geqslant P' \text{ s.t. } \mathtt{fv}\,Q \cap \overrightarrow{\beta^-} = \emptyset\}$ — $\mathtt{fv}\,\mathtt{nf}\,(Q) = \mathtt{fv}\,Q$ by Lemma 40

$= \{\mathtt{nf}\,(Q) \mid Q \in \mathsf{UB}(\Theta, \overrightarrow{\beta^-} \vdash P') \text{ s.t. } \mathtt{fv}\,Q \cap \overrightarrow{\beta^-} = \emptyset\}$ — by Lemma 77

$= \{\mathtt{nf}\,(Q) \mid Q \in \mathsf{UB}(\Theta \vdash \exists\overrightarrow{\beta^-}.\,P')\}$ — by the definition of UB

$= \{\mathtt{nf}\,(Q) \mid \Theta \vdash Q \geqslant \exists\overrightarrow{\beta^-}.\,P'\}$ — by Lemma 77

Case 3. $P = \downarrow M$ Let us prove the set equality by two inclusions.

$\subseteq$ Suppose that $\Theta \vdash Q \geqslant \downarrow M$ and M is normalized.

By Lemma 77, $Q \in \mathsf{UB}(\Theta \vdash \downarrow M)$. Then by definition of UB, $Q = \exists\overrightarrow{\alpha^-}.\,\downarrow M'$ for some $\overrightarrow{\alpha^-}$, M', and $\Theta \vdash \sigma : \overrightarrow{\alpha^-}$ s.t. $[\sigma]\downarrow M' \simeq^D \downarrow M$.

We need to show that $\mathtt{nf}\,(Q) \in \mathsf{NFUB}(\Theta \vdash \downarrow M)$. Notice that $\mathtt{nf}\,(Q) = \mathtt{nf}\,(\exists\overrightarrow{\alpha^-}.\,\downarrow M') = \exists\overrightarrow{\alpha^-}_0.\,\downarrow M_0$, where $\mathtt{nf}\,(M') = M_0$ and $\mathtt{ord}\,\overrightarrow{\alpha^-}\,\mathtt{in}\,M_0 = \overrightarrow{\alpha^-}_0$.

The belonging of $\exists\overrightarrow{\alpha^-}_0.\,\downarrow M_0$ to $\mathsf{NFUB}(\Theta \vdash \downarrow M)$ means that

1. $\mathtt{ord}\,\overrightarrow{\alpha^-}_0\,\mathtt{in}\,M_0 = \overrightarrow{\alpha^-}_0$ and
2. that there exists $\Theta \vdash \sigma_0 : \overrightarrow{\alpha^-}_0$ such that $[\sigma_0]\downarrow M_0 = \downarrow M$.

The first requirement holds by Corollary 13. To show the second requirement, we construct σ_0 as $\mathtt{nf}\,(\sigma|_{\mathtt{fv}\,M'})$. Let us show the required properties of σ_0:

1. $\Theta \vdash \sigma_0 : \overrightarrow{\alpha^-}_0$. Notice that by Lemma 7, $\Theta \vdash \sigma|_{\mathtt{fv}\,(M')} : \overrightarrow{\alpha^-} \cap \mathtt{fv}\,(M')$, which we rewrite as $\Theta \vdash \sigma|_{\mathtt{fv}\,(M')} : \overrightarrow{\alpha^-}_0$ (since by Lemma 35 $\overrightarrow{\alpha^-}_0 = \overrightarrow{\alpha^-} \cap \mathtt{fv}\,M_0$ as sets, and $\mathtt{fv}\,(M_0) = \mathtt{fv}\,(M')$ by Lemma 40). Then by Lemma 42, $\Theta \vdash \mathtt{nf}\,(\sigma|_{\mathtt{fv}\,(M')}) : \overrightarrow{\alpha^-}_0$, that is $\Theta \vdash \sigma_0 : \overrightarrow{\alpha^-}_0$.
2. $[\sigma_0]\downarrow M_0 = \downarrow M$. $[\sigma]\downarrow M' \simeq^D \downarrow M$ means $[\sigma|_{\mathtt{fv}\,(M')}]\downarrow M' \simeq^D \downarrow M$ by Lemma 6. Then by Lemma 45, $\mathtt{nf}\,([\sigma|_{\mathtt{fv}\,(M')}]\downarrow M') = \mathtt{nf}\,(\downarrow M)$, implying $[\sigma_0]\downarrow M_0 = \mathtt{nf}\,(\downarrow M)$ by Lemma 43, and further $[\sigma_0]\downarrow M_0 = \downarrow M$ by Lemma 46 (since $\downarrow M$ is normal by assumption).

$\supseteq$ Suppose that a type belongs to $\mathsf{NFUB}(\Theta \vdash \downarrow M)$ for a normalized $\downarrow M$. Then it must have shape $\exists\overrightarrow{\alpha^-}_0.\,\downarrow M_0$ for some $\overrightarrow{\alpha^-}_0$, M_0, and $\Theta \vdash \sigma_0 : \overrightarrow{\alpha^-}_0$ such that

$\mathtt{ord}\,\overrightarrow{\alpha^-}_0\,\mathtt{in}\,M_0 = \overrightarrow{\alpha^-}_0$ and $[\sigma_0]{\downarrow}M_0 = {\downarrow}M$. It suffices to show that 1. $\exists\overrightarrow{\alpha^-}_0.\,{\downarrow}M_0$ is normalized itself, and 2. $\Theta \vdash \exists\overrightarrow{\alpha^-}_0.\,{\downarrow}M_0 \geqslant {\downarrow}M$.

1. By definition, $\mathtt{nf}\,(\exists\overrightarrow{\alpha^-}_0.\,{\downarrow}M_0) = \exists\overrightarrow{\alpha^-}_1.\,{\downarrow}M_1$, where $M_1 = \mathtt{nf}\,(M_0)$ and $\mathtt{ord}\,\overrightarrow{\alpha^-}_0\,\mathtt{in}\,M_1 = \overrightarrow{\alpha^-}_1$. First, notice that by Lemmas 39 and 41, $\mathtt{ord}\,\overrightarrow{\alpha^-}_0\,\mathtt{in}\,M_1 = \mathtt{ord}\,\overrightarrow{\alpha^-}_0\,\mathtt{in}\,M_0 = \overrightarrow{\alpha^-}_0$. This way, $\mathtt{nf}\,(\exists\overrightarrow{\alpha^-}_0.\,{\downarrow}M_0) = \exists\overrightarrow{\alpha^-}_0.\,{\downarrow}\mathtt{nf}\,(M_0)$. Second, M_0 is normalized by Lemma 47, since $[\sigma_0]{\downarrow}M_0 = {\downarrow}M$ is normal. As such,
$\mathtt{nf}\,(\exists\overrightarrow{\alpha^-}_0.\,{\downarrow}M_0) = \exists\overrightarrow{\alpha^-}_0.\,{\downarrow}M_0$, in other words, $\exists\overrightarrow{\alpha^-}_0.\,{\downarrow}M_0$ is normalized.
2. $\Theta \vdash \exists\overrightarrow{\alpha^-}_0.\,{\downarrow}M_0 \geqslant {\downarrow}M$ holds immediately by $(\exists^{\geqslant})$ with the substitution σ_0. Notice that $\Theta \vdash [\sigma_0]{\downarrow}M_0 \geqslant {\downarrow}M$ follows from $[\sigma_0]{\downarrow}M_0 = {\downarrow}M$ by reflexivity of subtyping (Lemma 22).

■

Lemma 79. *Upper bounds of a type do not depend on the context as soon as the type is well-formed in it.*

If $\Theta_1 \vdash P$ *and* $\Theta_2 \vdash P$ *then* $\mathsf{UB}(\Theta_1 \vdash P) = \mathsf{UB}(\Theta_2 \vdash P)$ *and* $\mathsf{NFUB}(\Theta_1 \vdash P) = \mathsf{NFUB}(\Theta_2 \vdash P)$

Proof We prove both inclusions by structural induction on P.

Case 1. $P = \beta^+$ Then $\mathsf{UB}(\Theta_1 \vdash \beta^+) = \mathsf{UB}(\Theta_2 \vdash \beta^+) = \{\exists\overrightarrow{\alpha^-}.\,\beta^+ \mid \text{for some } \overrightarrow{\alpha^-}\}$. $\mathsf{NFUB}(\Theta_1 \vdash \beta^+) = \mathsf{NFUB}(\Theta_2 \vdash \beta^+) = \{\beta^+\}$.

Case 2. $P = \exists\overrightarrow{\beta^-}.\,P'$. Then $\mathsf{UB}(\Theta_1 \vdash \exists\overrightarrow{\beta^-}.\,P') = \mathsf{UB}(\Theta_1, \overrightarrow{\beta^-} \vdash P')$ not using $\overrightarrow{\beta^-}$. $\mathsf{UB}(\Theta_2 \vdash \exists\overrightarrow{\beta^-}.\,P') = \mathsf{UB}(\Theta_2, \overrightarrow{\beta^-} \vdash P')$ not using $\overrightarrow{\beta^-}$. By the induction hypothesis, $\mathsf{UB}(\Theta_1, \overrightarrow{\beta^-} \vdash P') = \mathsf{UB}(\Theta_2, \overrightarrow{\beta^-} \vdash P')$, and if we restrict these sets to the same domain, they stay equal. Analogously, $\mathsf{NFUB}(\Theta_1 \vdash \exists\overrightarrow{\beta^-}.\,P') = \mathsf{NFUB}(\Theta_2 \vdash \exists\overrightarrow{\beta^-}.\,P')$.

Case 3. $P = {\downarrow}M$. Suppose that $\exists\overrightarrow{\alpha^-}.\,{\downarrow}M' \in \mathsf{UB}(\Theta_1 \vdash {\downarrow}M)$. It means that $\Theta_1, \overrightarrow{\alpha^-} \vdash M'$ and there exist $\Theta_1 \vdash \vec{N}$ s.t. $[\vec{N}/\overrightarrow{\alpha^-}]{\downarrow}M' \simeq^D {\downarrow}M$, or in other terms, there exists $\Theta_1 \vdash \sigma : \overrightarrow{\alpha^-}$ such that $[\sigma]{\downarrow}M' \simeq^D {\downarrow}M$.

We need to show that $\exists\overrightarrow{\alpha^-}.\,{\downarrow}M' \in UB(\Theta_2 \vdash {\downarrow}M)$, in other words, $\Theta_2, \overrightarrow{\alpha^-} \vdash M'$ and there exists $\Theta_2 \vdash \sigma_0 : \overrightarrow{\alpha^-}$ such that $[\sigma_0]{\downarrow}M' \simeq^D {\downarrow}M$.

First, let us show $\Theta_2, \overrightarrow{\alpha^-} \vdash M'$. Notice that $[\sigma]{\downarrow}M' \simeq^D {\downarrow}M$ implies $\mathtt{fv}\,([\sigma]M') = \mathtt{fv}\,({\downarrow}M)$ by Lemma 26. By Lemma 15, $\mathtt{fv}\,(M') \setminus \overrightarrow{\alpha^-} \subseteq \mathtt{fv}\,([\sigma]M')$. This way, $\mathtt{fv}\,(M') \setminus \overrightarrow{\alpha^-} \subseteq \mathtt{fv}\,(M)$, implying $\mathtt{fv}\,(M') \subseteq \mathtt{fv}\,(M) \cup \overrightarrow{\alpha^-}$. By Lemma 3, $\Theta_2 \vdash {\downarrow}M$ implies $\mathtt{fv}\,M \subseteq \Theta_2$, hence, $\mathtt{fv}\,M' \subseteq (\Theta_2, \overrightarrow{\alpha^-})$, which by Corollary 1 means $\Theta_2, \overrightarrow{\alpha^-} \vdash M'$.

Second, let us construct the required σ_0 in the following way:

$$\begin{cases} [\sigma_0]\alpha_i^- = [\sigma]\alpha_i^- & \text{for } \alpha_i^- \in \overrightarrow{\alpha^-} \cap \mathtt{fv}\,(M') \\ [\sigma_0]\alpha_i^- = \forall\gamma^+.\uparrow\gamma^+ & \text{for } \alpha_i^- \in \overrightarrow{\alpha^-} \setminus \mathtt{fv}\,(M') \\ [\sigma_0]\gamma^{\pm} = \gamma^{\pm} & \text{for any other } \gamma^{\pm} \end{cases}$$

This construction of a substitution coincides with the one from the proof of Lemma 20. This way, for σ_0, hold the same properties:

1. $[\sigma_0]M' = [\sigma]M'$, which in particular, implies $[\sigma_0]{\downarrow}M = [\sigma]{\downarrow}M$, and thus, $[\sigma]{\downarrow}M' \simeq^D {\downarrow}M$ can be rewritten to $[\sigma_0]{\downarrow}M' \simeq^D {\downarrow}M$; and
2. $\mathtt{fv}\,([\sigma]M') \vdash \sigma_0 : \overrightarrow{\alpha^-}$, which, as noted above, can be rewritten to $\mathtt{fv}\,(M) \vdash \sigma_0 : \overrightarrow{\alpha^-}$, and since $\mathtt{fv}\,M \subseteq \Theta_2$, weakened to $\Theta_2 \vdash \sigma_0 : \overrightarrow{\alpha^-}$.

The proof of $\mathsf{NFUB}(\Theta_1 \vdash {\downarrow}M) \subseteq \mathsf{NFUB}(\Theta_2 \vdash {\downarrow}M)$ is analogous. The differences are:

1. $\mathtt{ord}\,\overrightarrow{\alpha^-}\,\mathtt{in}\,M' = \overrightarrow{\alpha^-}$ holds by assumption,
2. $[\sigma]{\downarrow}M' = {\downarrow}M$ implies $\mathtt{fv}\,([\sigma]M') = \mathtt{fv}\,({\downarrow}M)$ by rewriting,
3. $[\sigma]{\downarrow}M' = {\downarrow}M$ and $[\sigma_0]{\downarrow}M = [\sigma]{\downarrow}M$ imply $[\sigma_0]{\downarrow}M' = {\downarrow}M$ by rewriting.

■

Lemma 80 (Soundness of the Least Upper Bound)**.** *For types* $\Theta \vdash P_1$*, and* $\Theta \vdash P_2$*, if* $\Theta \vDash P_1 \vee P_2 = Q$ *then*

(i) $\Theta \vdash Q$

(ii) $\Theta \vdash Q \geqslant P_1$ *and* $\Theta \vdash Q \geqslant P_2$

Proof Induction on $\Theta \vDash P_1 \vee P_2 = Q$.

Case 1. $\Theta \vDash \alpha^+ \vee \alpha^+ = \alpha^+$
Then $\Theta \vdash \alpha^+$ by assumption, and $\Theta \vdash \alpha^+ \geqslant \alpha^+$ by $(\textsc{Var}^{\geqslant}_{+})$.

Case 2. $\Theta \vDash \exists\overrightarrow{\alpha^-}.\, P_1 \vee \exists\overrightarrow{\beta^-}.\, P_2 = Q$
Then by inversion of $\Theta \vdash \exists\overrightarrow{\alpha^-}.\, P_i$ and weakening, $\Theta, \overrightarrow{\alpha^-}, \overrightarrow{\beta^-} \vdash P_i$, hence, the induction hypothesis applies to $\Theta, \overrightarrow{\alpha^-}, \overrightarrow{\beta^-} \vDash P_1 \vee P_2 = Q$. Then

(i) $\Theta, \overrightarrow{\alpha^-}, \overrightarrow{\beta^-} \vdash Q$,

(ii) $\Theta, \overrightarrow{\alpha^-}, \overrightarrow{\beta^-} \vdash Q \geqslant P_1$,

(iii) $\Theta, \overrightarrow{\alpha^-}, \overrightarrow{\beta^-} \vdash Q \geqslant P_2$.

To prove $\Theta \vdash Q$, it suffices to show that $\mathtt{fv}\,(Q) \cap (\Theta, \overrightarrow{\alpha^-}, \overrightarrow{\beta^-}) = \mathtt{fv}\,(Q) \cap \Theta$ (and then apply Lemma 4). The inclusion right-to-left is self-evident. To show $\mathtt{fv}\,(Q) \cap (\Theta, \overrightarrow{\alpha^-}, \overrightarrow{\beta^-}) \subseteq \mathtt{fv}\,(Q) \cap \Theta$, we prove that $\mathtt{fv}\,(Q) \subseteq \Theta$.

$$
\begin{aligned}
\mathtt{fv}\,(Q) &\subseteq \mathtt{fv}\,P_1 \cap \mathtt{fv}\,P_2 && \text{by Lemma 17}\\
&\subseteq ((\Theta, \overrightarrow{\alpha^-}) \setminus \overrightarrow{\beta^-}) \cap ((\Theta, \overrightarrow{\beta^-}) \setminus \overrightarrow{\alpha^-}) && \text{since } \Theta \vdash \exists\overrightarrow{\alpha^-}.\, P_1,\ \mathtt{fv}\,(P_1) \subseteq (\Theta, \overrightarrow{\alpha^-}) = (\Theta, \overrightarrow{\alpha^-}) \setminus \overrightarrow{\beta^-} \text{(the latter is because by the Barendregt's convention, } (\Theta, \overrightarrow{\alpha^-}) \cap \overrightarrow{\beta^-} = \emptyset)\\
& && \text{similarly, } \mathtt{fv}\,(P_2) \subseteq (\Theta, \overrightarrow{\beta^-}) \setminus \overrightarrow{\alpha^-}\\
&\subseteq \Theta
\end{aligned}
$$

To show $\Theta \vdash Q \geqslant \exists \overrightarrow{\alpha}.\, P_1$, we apply $(\exists^{\geqslant})$. Then $\Theta, \overrightarrow{\alpha} \vdash Q \geqslant P_1$ holds since $\Theta, \overrightarrow{\alpha}, \overrightarrow{\beta} \vdash Q \geqslant P_1$ (by the induction hypothesis), $\Theta, \overrightarrow{\alpha} \vdash Q$ (by weakening), and $\Theta, \overrightarrow{\alpha} \vdash P_1$.

Judgment $\Theta \vdash Q \geqslant \exists \overrightarrow{\beta}.\, P_2$ is proved symmetrically.

Case 3. $\Theta \vDash \downarrow N \vee \downarrow M = \exists \overrightarrow{\alpha}.\, [\overrightarrow{\alpha}/\widehat{\Theta}]P$. By the inversion, $\Theta, \cdot \vDash \mathtt{nf}\,(\downarrow N) \overset{a}{\simeq} \mathtt{nf}\,(\downarrow M) \dashv (\widehat{\Theta}, P, \widehat{\tau}_1, \widehat{\tau}_2)$. Then by the soundness of anti-unification (Lemma 74),

(i) $\Theta\,;\widehat{\Theta} \vdash P$, then by Lemma 61,

$$\Theta, \overrightarrow{\alpha} \vdash [\overrightarrow{\alpha}/\widehat{\Theta}]P \tag{C.7}$$

(ii) $\Theta\,;\cdot \vdash \widehat{\tau}_1 : \widehat{\Theta}$ and $\Theta\,;\cdot \vdash \widehat{\tau}_2 : \widehat{\Theta}$. Assuming that $\widehat{\Theta} = \widehat{\beta}_1^{-}, \ldots, \widehat{\beta}_n^{-}$, the antiunification solutions $\widehat{\tau}_1$ and $\widehat{\tau}_2$ can be put explicitly as $\widehat{\tau}_1 = (\widehat{\beta}_1^{-} :\simeq N_1, \ldots, \widehat{\beta}_n^{-} :\simeq N_n)$, and $\widehat{\tau}_2 = (\widehat{\beta}_1^{-} :\simeq M_1, \ldots, \widehat{\beta}_n^{-} :\simeq M_n)$. Then

$$\widehat{\tau}_1 = (\overrightarrow{N}/\overrightarrow{\alpha}) \circ (\overrightarrow{\alpha}/\widehat{\Theta}) \tag{C.8}$$

$$\widehat{\tau}_2 = (\overrightarrow{M}/\overrightarrow{\alpha}) \circ (\overrightarrow{\alpha}/\widehat{\Theta}) \tag{C.9}$$

(iii) $[\widehat{\tau}_1]Q = P_1$ and $[\widehat{\tau}_2]Q = P_1$, which, by C.8 and C.9, means

$$[\overrightarrow{N}/\overrightarrow{\alpha}][\overrightarrow{\alpha}/\widehat{\Theta}]P = \mathtt{nf}\,(\downarrow N) \tag{C.10}$$

$$[\overrightarrow{M}/\overrightarrow{\alpha}][\overrightarrow{\alpha}/\widehat{\Theta}]P = \mathtt{nf}\,(\downarrow M) \tag{C.11}$$

Then $\Theta \vdash \exists \overrightarrow{\alpha}.\, [\overrightarrow{\alpha}/\widehat{\Theta}]P$ follows directly from C.7.

To show $\Theta \vdash \exists \overrightarrow{\alpha}.\, [\overrightarrow{\alpha}/\widehat{\Theta}]P \geqslant \downarrow N$, we apply $(\exists^{\geqslant})$, instantiating $\overrightarrow{\alpha}$ with $\overrightarrow{N}$. Then $\Theta \vdash [\overrightarrow{N}/\overrightarrow{\alpha}][\overrightarrow{\alpha}/\widehat{\Theta}]P \geqslant \downarrow N$ follows from C.10 and since $\Theta \vdash \mathtt{nf}\,(\downarrow N) \geqslant \downarrow N$ (by Corollary 16).

Analogously, instantiating $\overrightarrow{\alpha}$ with $\overrightarrow{M}$, gives us $\Theta \vdash [\overrightarrow{M}/\overrightarrow{\alpha}][\overrightarrow{\alpha}/\widehat{\Theta}]P \geqslant \downarrow M$ (from C.11), and hence, $\Theta \vdash \exists \overrightarrow{\alpha}.\, [\overrightarrow{\alpha}/\widehat{\Theta}]P \geqslant \downarrow M$.

■

Lemma 81 (Completeness and Initiality of the Least Upper Bound)**.** *For types $\Theta \vdash P_1$, $\Theta \vdash P_2$, and $\Theta \vdash Q$ such that $\Theta \vdash Q \geqslant P_1$ and $\Theta \vdash Q \geqslant P_2$, there exists Q' s.t. $\Theta \vDash P_1 \vee P_2 = Q'$ and $\Theta \vdash Q \geqslant Q'$.*

Proof Induction on the pair (P_1, P_2). From Lemma 78, $Q \in \mathsf{UB}(\Theta \vdash P_1) \cap \mathsf{UB}(\Theta \vdash P_2)$. Let us consider the cases of what P_1 and P_2 are (i.e. the last rules to infer $\Theta \vdash P_i$).

Case 1. $P_1 = \exists \overrightarrow{\beta}_1.\, Q_1$, $P_2 = \exists \overrightarrow{\beta}_2.\, Q_2$, where either $\overrightarrow{\beta}_1$ or $\overrightarrow{\beta}_2$ is not empty

Then

$$\begin{aligned} Q &\in \mathsf{UB}(\Theta \vdash \exists \overrightarrow{\beta}_1.\, Q_1) \cap \mathsf{UB}(\Theta \vdash \exists \overrightarrow{\beta}_2.\, Q_2) \\ &\subseteq \mathsf{UB}(\Theta, \overrightarrow{\beta}_1 \vdash Q_1) \cap \mathsf{UB}(\Theta, \overrightarrow{\beta}_2 \vdash Q_2) && \text{definition of UB} \end{aligned}$$

$$= \mathsf{UB}(\Theta, \overrightarrow{\beta^-}_1, \overrightarrow{\beta^-}_2 \vdash Q_1) \cap \mathsf{UB}(\Theta, \overrightarrow{\beta^-}_1, \overrightarrow{\beta^-}_2 \vdash Q_2) \qquad \text{by Lemma 79}$$
$$= \{Q' \mid \Theta, \overrightarrow{\beta^-}_1, \overrightarrow{\beta^-}_2 \vdash Q' \geqslant Q_1\} \cap \{Q' \mid \Theta, \overrightarrow{\beta^-}_1, \overrightarrow{\beta^-}_2 \vdash Q' \geqslant Q_2\} \qquad \text{by Lemma 77}$$

It means that $\Theta, \overrightarrow{\beta^-}_1, \overrightarrow{\beta^-}_2 \vdash Q \geqslant Q_1$ and $\Theta, \overrightarrow{\beta^-}_1, \overrightarrow{\beta^-}_2 \vdash Q \geqslant Q_2$. Then the next step of the algorithm—the recursive call $\Theta, \overrightarrow{\beta^-}_1, \overrightarrow{\beta^-}_2 \vdash Q_1 \vee Q_2 = Q'$ terminates by the induction hypothesis, and moreover, $\Theta, \overrightarrow{\beta^-}_1, \overrightarrow{\beta^-}_2 \vdash Q \geqslant Q'$. This way, the result of the algorithm is Q', i.e. $\Theta \vdash P_1 \vee P_2 = Q'$.

Since both Q and Q' are sound upper bounds, $\Theta \vdash Q$ and $\Theta \vdash Q'$, and therefore, $\Theta, \overrightarrow{\beta^-}_1, \overrightarrow{\beta^-}_2 \vdash Q \geqslant Q'$ can be strengthened to $\Theta \vdash Q \geqslant Q'$ by Lemma 20.

Case 2. $P_1 = \alpha^+$ and $P_2 = {\downarrow}N$

Then the set of common upper bounds of ${\downarrow}N$ and α^+ is empty, and thus, $Q \in \mathsf{UB}(\Theta \vdash P_1) \cap \mathsf{UB}(\Theta \vdash P_2)$ gives a contradiction:

$$\begin{aligned} & Q \in \mathsf{UB}(\Theta \vdash \alpha^+) \cap \mathsf{UB}(\Theta \vdash {\downarrow}N) \\ & = \{\exists \overrightarrow{\alpha^-}.\ \alpha^+ \mid \cdots\} \cap \{\exists \overrightarrow{\beta^-}.\ {\downarrow}M' \mid \cdots\} && \text{by the definition of } \mathsf{UB} \\ & = \emptyset && \text{since } \alpha^+ \neq {\downarrow}M' \text{ for any } M' \end{aligned}$$

Case 3. $P_1 = {\downarrow}N$ and $P_2 = \alpha^+$

Symmetric to case 2

Case 4. $P_1 = \alpha^+$ and $P_2 = \beta^+$ (where $\beta^+ \neq \alpha^+$)

Similarly to case 2, the set of common upper bounds is empty, which leads to the contradiction:

$$\begin{aligned} & Q \in \mathsf{UB}(\Theta \vdash \alpha^+) \cap \mathsf{UB}(\Theta \vdash \beta^+) \\ & = \{\exists \overrightarrow{\alpha^-}.\ \alpha^+ \mid \cdots\} \cap \{\exists \overrightarrow{\beta^-}.\ \beta^+ \mid \cdots\} && \text{by the definition of } \mathsf{UB} \\ & = \emptyset && \text{since } \alpha^+ \neq \beta^+ \end{aligned}$$

Case 5. $P_1 = \alpha^+$ and $P_2 = \alpha^+$

Then the algorithm terminates in one step ($(\textsc{Var}^\vee)$) and the result is α^+, i.e. $\Theta \vdash \alpha^+ \vee \alpha^+ = \alpha^+$.

Since $Q \in \mathsf{UB}(\Theta \vdash \alpha^+)$, $Q = \exists \overrightarrow{\alpha^-}.\ \alpha^+$. Then $\Theta \vdash \exists \overrightarrow{\alpha^-}.\ \alpha^+ \geqslant \alpha^+$ by $(\exists^\geqslant)$: $\overrightarrow{\alpha^-}$ can be instantiated with arbitrary negative types (for example $\forall\beta^+.\ {\uparrow}\beta^+$), since the substitution for unused variables does not change the term $[\overrightarrow{N}/\overrightarrow{\alpha^-}]\alpha^+ = \alpha^+$, and then $\Theta \vdash \alpha^+ \geqslant \alpha^+$ by $(\textsc{Var}^\geqslant_+)$.

Case 6. $P_1 = {\downarrow}M_1$ and $P_2 = {\downarrow}M_2$

In the next step, the algorithm tries to anti-unify $\mathtt{nf}\,({\downarrow}M_1)$ and $\mathtt{nf}\,({\downarrow}M_2)$. By Lemma 75, to show that the anti-unification algorithm terminates, it suffices to demonstrate that a sound anti-unification solution exists. Notice that

$$\mathtt{nf}\,(Q) \in \mathsf{NFUB}(\Theta \vdash \mathtt{nf}\,({\downarrow}M_1)) \cap \mathsf{NFUB}(\Theta \vdash \mathtt{nf}\,({\downarrow}M_2))$$

$$= \mathsf{NFUB}(\Theta \vdash \downarrow \mathtt{nf}\,(M_1)) \cap \mathsf{NFUB}(\Theta \vdash \downarrow \mathtt{nf}\,(M_2))$$

$$= \left\{ \exists \overrightarrow{\alpha^-}.\downarrow M' \;\middle|\; \begin{array}{l} \text{for } \overrightarrow{\alpha^-}, M', \text{ and } \vec{N} \text{ s.t. } \mathtt{ord}\,\overrightarrow{\alpha^-}\,\mathtt{in}\,M' = \overrightarrow{\alpha^-}, \\ \Theta \vdash N_i, \Theta, \overrightarrow{\alpha^-} \vdash M', \text{ and } [\vec{N}/\overrightarrow{\alpha^-}]\downarrow M' = \downarrow \mathtt{nf}\,(M_1) \end{array} \right\}$$

$$= \cap \left\{ \exists \overrightarrow{\alpha^-}.\downarrow M' \;\middle|\; \begin{array}{l} \text{for } \overrightarrow{\alpha^-}, M', \text{ and } \vec{N} \text{ s.t. } \mathtt{ord}\,\overrightarrow{\alpha^-}\,\mathtt{in}\,M' = \overrightarrow{\alpha^-}, \\ \Theta \vdash \vec{N}_1, \Theta \vdash \vec{N}_2, \Theta, \overrightarrow{\alpha^-} \vdash M', \text{ and } [\vec{N}/\overrightarrow{\alpha^-}]\downarrow M' = \downarrow \mathtt{nf}\,(M_2) \end{array} \right\}$$

$$= \left\{ \exists \overrightarrow{\alpha^-}.\downarrow M' \;\middle|\; \begin{array}{l} \text{for } \overrightarrow{\alpha^-}, M', \vec{N}_1 \text{ and } \vec{N}_2 \text{ s.t. } \mathtt{ord}\,\overrightarrow{\alpha^-}\,\mathtt{in}\,M' = \overrightarrow{\alpha^-}, \\ \Theta \vdash \vec{N}_1, \Theta \vdash \vec{N}_2, \Theta, \overrightarrow{\alpha^-} \vdash M', [\vec{N}_1/\overrightarrow{\alpha^-}]\downarrow M' = \downarrow \mathtt{nf}\,(M_1) \\ \text{, and } [\vec{N}_2/\overrightarrow{\alpha^-}]\downarrow M' = \downarrow \mathtt{nf}\,(M_2) \end{array} \right\}$$

The fact that the latter set is non-empty means that there exist $\overrightarrow{\alpha^-}$, M', $\vec{N}_1$ and $\vec{N}_2$ such that

(i) $\Theta, \overrightarrow{\alpha^-} \vdash M'$ (notice that M' is normal)

(ii) $\Theta \vdash \vec{N}_1$ and $\Theta \vdash \vec{N}_1$,

(iii) $[\vec{N}_1/\overrightarrow{\alpha^-}]\downarrow M' = \downarrow \mathtt{nf}\,(M_1)$ and $[\vec{N}_2/\overrightarrow{\alpha^-}]\downarrow M' = \downarrow \mathtt{nf}\,(M_2)$

For each negative variable α^- from $\overrightarrow{\alpha^-}$, let us choose a fresh negative anti-unification variable $\widehat{\alpha}^-$, and denote the list of these variables as $\overrightarrow{\widehat{\alpha}^-}$. Let us show that $(\overrightarrow{\widehat{\alpha}^-},\ [\overrightarrow{\widehat{\alpha}^-}/\overrightarrow{\alpha^-}]\downarrow M',\ \vec{N}_1/\overrightarrow{\widehat{\alpha}^-},\ \vec{N}_2/\overrightarrow{\widehat{\alpha}^-})$ is a sound anti-unifier of $\mathtt{nf}\,(\downarrow M_1)$ and $\mathtt{nf}\,(\downarrow M_2)$ in context Θ:

- $\overrightarrow{\widehat{\alpha}^-}$ is negative by construction,
- $\Theta\,;\overrightarrow{\widehat{\alpha}^-} \vdash [\overrightarrow{\widehat{\alpha}^-}/\overrightarrow{\alpha^-}]\downarrow M'$ because $\Theta, \overrightarrow{\alpha^-} \vdash \downarrow M'$ (Lemma 60),
- $\Theta\,;\cdot \vdash (\vec{N}_1/\overrightarrow{\widehat{\alpha}^-}) : \overrightarrow{\widehat{\alpha}^-}$ because $\Theta \vdash \vec{N}_1$ and $\Theta\,;\cdot \vdash (\vec{N}_2/\overrightarrow{\widehat{\alpha}^-}) : \overrightarrow{\widehat{\alpha}^-}$ because $\Theta \vdash \vec{N}_2$,
- $[\vec{N}_1/\overrightarrow{\widehat{\alpha}^-}][\overrightarrow{\widehat{\alpha}^-}/\overrightarrow{\alpha^-}]\downarrow M' = [\vec{N}_1/\overrightarrow{\alpha^-}]\downarrow M' = \downarrow \mathtt{nf}\,(M_1) = \mathtt{nf}\,(\downarrow M_1)$.
- $[\vec{N}_2/\overrightarrow{\widehat{\alpha}^-}][\overrightarrow{\widehat{\alpha}^-}/\overrightarrow{\alpha^-}]\downarrow M' = [\vec{N}_2/\overrightarrow{\alpha^-}]\downarrow M' = \downarrow \mathtt{nf}\,(M_2) = \mathtt{nf}\,(\downarrow M_2)$.

Then by the completeness of the anti-unification (Lemma 75), the anti-unification algorithm terminates, so is the Least Upper Bound algorithm invoking it, i.e. $Q' = \exists \overrightarrow{\beta^-}.\,[\overrightarrow{\beta^-}/\widehat{\Theta}]P$, where $(\widehat{\Theta}, P, \widehat{\tau}_1, \widehat{\tau}_2)$ is the result of the anti-unification of $\mathtt{nf}\,(\downarrow M_1)$ and $\mathtt{nf}\,(\downarrow M_2)$ in context Θ.

Moreover, Lemma 75 also says that the found anti-unification solution is initial, i.e. there exists $\widehat{\tau}$ such that $\Theta\,;\widehat{\Theta} \vdash \widehat{\tau} : \overrightarrow{\widehat{\alpha}^-}$ and $[\widehat{\tau}][\overrightarrow{\widehat{\alpha}^-}/\overrightarrow{\alpha^-}]\downarrow M' = P$.

Let σ be a sequential Kleisli composition of the following substitutions: (i) $\overrightarrow{\widehat{\alpha}^-}/\overrightarrow{\alpha^-}$, (ii) $\widehat{\tau}$, and (iii) $\overrightarrow{\beta^-}/\widehat{\Theta}$. Notice that $\Theta, \overrightarrow{\beta^-} \vdash \sigma : \overrightarrow{\alpha^-}$ and $[\sigma]\downarrow M' = [\overrightarrow{\beta^-}/\widehat{\Theta}][\widehat{\tau}][\overrightarrow{\widehat{\alpha}^-}/\overrightarrow{\alpha^-}]\downarrow M' = [\overrightarrow{\beta^-}/\widehat{\Theta}]P$. In particular, from the reflexivity of subtyping: $\Theta, \overrightarrow{\beta^-} \vdash [\sigma]\downarrow M' \geqslant [\overrightarrow{\beta^-}/\widehat{\Theta}]P$.

It allows us to show $\Theta \vdash \mathtt{nf}\,(Q) \geqslant Q'$, i.e. $\Theta \vdash \exists \overrightarrow{\alpha^-}.\downarrow M' \geqslant \exists \overrightarrow{\beta^-}.\,[\overrightarrow{\beta^-}/\widehat{\Theta}]P$, by applying $(\exists^{\geqslant})$, instantiating $\overrightarrow{\alpha^-}$ with respect to σ. Finally, $\Theta \vdash Q \geqslant Q'$ by transitively combining $\Theta \vdash \mathtt{nf}\,(Q) \geqslant Q'$ and $\Theta \vdash Q \geqslant \mathtt{nf}\,(Q)$ (holds by Corollary 16 and inversion).

■

C.3.9 Upgrade

Let us consider a type P well-formed in Θ. Some of its Θ-supertypes are also well-formed in a smaller context $\Theta_0 \subseteq \Theta$. The upgrade is the operation that returns the least of such supertypes.

Observation 11 (Upgrade determinism)**.** *Assuming P is well-formed in $\Theta \subseteq \Theta_0$, if* $\texttt{upgrade}\ \Theta \vdash P\ \texttt{to}\ \Theta_0 = Q$ *and* $\texttt{upgrade}\ \Theta \vdash P\ \texttt{to}\ \Theta_0 = Q'$ *are defined then $Q = Q'$.*

Proof It follows directly from Observation 10, and the convention that the fresh variables are chosen by a fixed deterministic algorithm (Section A.2.2). ■

Lemma 82 (Soundness of Upgrade)**.** *Assuming P is well-formed in $\Theta = \Theta_0, \overrightarrow{\alpha^{\pm}}$, if* $\texttt{upgrade}\ \Theta \vdash P\ \texttt{to}\ \Theta_0 = Q$ *then*

1. $\Theta_0 \vdash Q$
2. $\Theta \vdash Q \geqslant P$

Lemma 103 (Soundness of Upgrade)**.** *Assuming P is well-formed in $\Theta = \Theta_0, \overrightarrow{\alpha^{\pm}}$, if* $\texttt{upgrade}\ \Theta \vdash P\ \texttt{to}\ \Theta_0 = Q$ *then*

1. $\Theta_0 \vdash Q$
2. $\Theta \vdash Q \geqslant P$

Proof By inversion, $\texttt{upgrade}\ \Theta \vdash P\ \texttt{to}\ \Theta_0 = Q$ means that for fresh $\overrightarrow{\beta^{\pm}}$ and $\overrightarrow{\gamma^{\pm}}$, $\Theta_0, \overrightarrow{\beta^{\pm}}, \overrightarrow{\gamma^{\pm}} \vdash [\overrightarrow{\beta^{\pm}}/\overrightarrow{\alpha^{\pm}}]P \vee [\overrightarrow{\gamma^{\pm}}/\overrightarrow{\alpha^{\pm}}]P = Q$. Then by the soundness of the least upper bound (Lemma 80),

1. $\Theta_0, \overrightarrow{\beta^{\pm}}, \overrightarrow{\gamma^{\pm}} \vdash Q$,
2. $\Theta_0, \overrightarrow{\beta^{\pm}}, \overrightarrow{\gamma^{\pm}} \vdash Q \geqslant [\overrightarrow{\beta^{\pm}}/\overrightarrow{\alpha^{\pm}}]P$, and
3. $\Theta_0, \overrightarrow{\beta^{\pm}}, \overrightarrow{\gamma^{\pm}} \vdash Q \geqslant [\overrightarrow{\gamma^{\pm}}/\overrightarrow{\alpha^{\pm}}]P$.

$$
\begin{aligned}
&\texttt{fv}\ Q \subseteq \texttt{fv}\ [\overrightarrow{\beta^{\pm}}/\overrightarrow{\alpha^{\pm}}]P \cap \texttt{fv}\ [\overrightarrow{\gamma^{\pm}}/\overrightarrow{\alpha^{\pm}}]P && \text{since by Lemma 17, } \texttt{fv}\ Q \subseteq \texttt{fv}\ [\overrightarrow{\beta^{\pm}}/\overrightarrow{\alpha^{\pm}}]P \\
& && \text{and } \texttt{fv}\ Q \subseteq \texttt{fv}\ [\overrightarrow{\gamma^{\pm}}/\overrightarrow{\alpha^{\pm}}]P \\
&\quad \subseteq ((\texttt{fv}\ P \setminus \overrightarrow{\alpha^{\pm}}) \cup \overrightarrow{\beta^{\pm}}) \cap ((\texttt{fv}\ P \setminus \overrightarrow{\alpha^{\pm}}) \cup \overrightarrow{\gamma^{\pm}}) && \\
&\quad = (\texttt{fv}\ P \setminus \overrightarrow{\alpha^{\pm}}) \cap (\texttt{fv}\ P \setminus \overrightarrow{\alpha^{\pm}}) && \text{since } \overrightarrow{\beta^{\pm}} \text{ and } \overrightarrow{\gamma^{\pm}} \text{ are fresh} \\
&\quad = \texttt{fv}\ P \setminus \overrightarrow{\alpha^{\pm}} && \\
&\quad \subseteq \Theta \setminus \overrightarrow{\alpha^{\pm}} && \text{since } P \text{ is well-formed in } \Theta \\
&\quad \subseteq \Theta_0 &&
\end{aligned}
$$

This way, by Lemma 4, $\Theta_0 \vdash Q$.

Let us apply $\overrightarrow{\alpha^{\pm}}/\overrightarrow{\beta^{\pm}}$—the inverse of the substitution $\overrightarrow{\beta^{\pm}}/\overrightarrow{\alpha^{\pm}}$ to both sides of $\Theta_0, \overrightarrow{\beta^{\pm}}, \overrightarrow{\gamma^{\pm}} \vdash Q \geqslant [\overrightarrow{\beta^{\pm}}/\overrightarrow{\alpha^{\pm}}]P$ and by Lemma 23 (since $\overrightarrow{\beta^{\pm}}/\overrightarrow{\alpha^{\pm}}$ can be specified as $\Theta_0, \overrightarrow{\beta^{\pm}}, \overrightarrow{\gamma^{\pm}} \vdash \overrightarrow{\beta^{\pm}}/\overrightarrow{\alpha^{\pm}} : \Theta_0, \overrightarrow{\alpha^{\pm}}, \overrightarrow{\gamma^{\pm}}$ by Lemma 14) obtain $\Theta_0, \overrightarrow{\alpha^{\pm}}, \overrightarrow{\gamma^{\pm}} \vdash [\overrightarrow{\alpha^{\pm}}/\overrightarrow{\beta^{\pm}}]Q \geqslant P$. Notice that $\Theta_0 \vdash Q$ implies

that $\mathtt{fv}\ Q \cap \overrightarrow{\beta^{\pm}} = \emptyset$, then by Corollary 3, $[\overrightarrow{\alpha^{\pm}}/\overrightarrow{\beta^{\pm}}]Q = Q$, and thus $\Theta_0, \overrightarrow{\alpha^{\pm}}, \overrightarrow{\gamma^{\pm}} \vdash Q \geqslant P$. By context strengthening, $\Theta_0, \overrightarrow{\alpha^{\pm}} \vdash Q \geqslant P$. ■

Lemma 83 (Completeness and Initiality of Upgrade)**.** *The upgrade returns the least* Θ*-supertype of* P *well-formed in* Θ_0*. Assuming* P *is well-formed in* $\Theta = \Theta_0, \overrightarrow{\alpha^{\pm}}$*,*
For any Q' *such that*

1. $\Theta_0 \vdash Q'$ *and*
2. $\Theta \vdash Q' \geqslant P$*,*

the result of the upgrade algorithm Q *exists (*$\mathtt{upgrade}\ \Theta \vdash P\ \mathtt{to}\ \Theta_0 = Q$*) and satisfies* $\Theta_0 \vdash Q' \geqslant Q$*.*

Proof Let us consider fresh (not intersecting with Θ) $\overrightarrow{\beta^{\pm}}$ and $\overrightarrow{\gamma^{\pm}}$.

If we apply substitution $\overrightarrow{\beta^{\pm}}/\overrightarrow{\alpha^{\pm}}$ to both sides of $\Theta_0, \overrightarrow{\alpha^{\pm}} \vdash Q' \geqslant P$, we have $\Theta_0, \overrightarrow{\beta^{\pm}} \vdash [\overrightarrow{\beta^{\pm}}/\overrightarrow{\alpha^{\pm}}]Q' \geqslant [\overrightarrow{\beta^{\pm}}/\overrightarrow{\alpha^{\pm}}]P$, which by Corollary 3, since $\overrightarrow{\alpha^{\pm}}$ is disjoint from $\mathtt{fv}\ (Q')$ (because $\Theta_0 \vdash Q'$), simplifies to $\Theta_0, \overrightarrow{\beta^{\pm}} \vdash Q' \geqslant [\overrightarrow{\beta^{\pm}}/\overrightarrow{\alpha^{\pm}}]P$.

Analogously, if we apply substitution $\overrightarrow{\gamma^{\pm}}/\overrightarrow{\alpha^{\pm}}$ to both sides of $\Theta_0, \overrightarrow{\alpha^{\pm}} \vdash Q' \geqslant P$, we have $\Theta_0, \overrightarrow{\gamma^{\pm}} \vdash Q' \geqslant [\overrightarrow{\gamma^{\pm}}/\overrightarrow{\alpha^{\pm}}]P$.

This way, Q' is a common supertype of $[\overrightarrow{\beta^{\pm}}/\overrightarrow{\alpha^{\pm}}]P$ and $[\overrightarrow{\gamma^{\pm}}/\overrightarrow{\alpha^{\pm}}]P$ in context $\Theta_0, \overrightarrow{\beta^{\pm}}, \overrightarrow{\gamma^{\pm}}$. It means that we can apply the completeness of the least upper bound (Lemma 81):

1. there exists Q s.t. $\Theta \vDash [\overrightarrow{\beta^{\pm}}/\overrightarrow{\alpha^{\pm}}]P \vee [\overrightarrow{\gamma^{\pm}}/\overrightarrow{\alpha^{\pm}}]P = Q$
2. $\Theta \vdash Q' \geqslant Q$.

The former means that the upgrade algorithm terminates and returns Q. The latter means that since both Q' and Q are well-formed in Θ_0 and Θ, by Lemma 20, $\Theta_0 \vdash Q' \geqslant Q$. ■

C.3.10 Constraint Satisfaction

Lemma 84 (Any constraint is satisfiable)**.** *Suppose that* $\Xi \vdash C$ *and* $\widehat{\Theta}$ *is a set such that* $\mathtt{dom}\ (C) \subseteq \widehat{\Theta} \subseteq \mathtt{dom}\ (\Xi)$*. Then there exists* $\widehat{\sigma}$ *such that* $\Xi \vdash \widehat{\sigma} : \widehat{\Theta}$ *and* $\Xi \vdash \widehat{\sigma} : C$*.*

Proof Let us define $\widehat{\sigma}$ on $\mathtt{dom}\ (C)$ in the following way:

$$[\widehat{\sigma}]\widehat{\alpha}^{\pm} = \begin{cases} P & \text{if } (\widehat{\alpha}^{\pm} :\simeq P) \in C \\ P & \text{if } (\widehat{\alpha}^{\pm} :\geqslant P) \in C \\ N & \text{if } (\widehat{\alpha}^{\pm} :\simeq N) \in C \\ \exists \beta^{-}.\downarrow\beta^{-} & \text{if } \widehat{\alpha}^{\pm} = \widehat{\alpha}^{+} \in \widehat{\Theta} \setminus \mathtt{dom}\ (C) \\ \forall \beta^{+}.\uparrow\beta^{+} & \text{if } \widehat{\alpha}^{\pm} = \widehat{\alpha}^{-} \in \widehat{\Theta} \setminus \mathtt{dom}\ (C) \end{cases}$$

Then $\Xi \vdash \widehat{\sigma} : C$ follows immediately from the reflexivity of equivalence and subtyping (Lemma 22) and the corresponding rules $(:\simeq^{\mathrm{SAT}}_{+})$, $(:\simeq^{\mathrm{SAT}}_{-})$, and $(:\geqslant^{\mathrm{SAT}}_{+})$. ■

Lemma 85 (Constraint Entry Satisfaction is Stable under Equivalence)**.**

– *If* $\Theta \vdash N_1 : e$ *and* $\Theta \vdash N_1 \simeq^{\leqslant} N_2$ *then* $\Theta \vdash N_2 : e$*.*

+ *If* $\Theta \vdash P_1 : e$ *and* $\Theta \vdash P_1 \simeq^{\leqslant} P_2$ *then* $\Theta \vdash P_2 : e$.

Proof

– Then e has form $(\widehat{\alpha}^- :\simeq M)$, and by inversion, $\Theta \vdash N_1 \simeq^{\leqslant} M$. Then by transitivity, $\Theta \vdash N_2 \simeq^{\leqslant} M$, meaning $\Theta \vdash N_2 : e$.

+ Let us consider what form e has.

Case 1. $e = (\widehat{\alpha}^+ :\simeq Q)$. Then $\Theta \vdash P_1 \simeq^{\leqslant} Q$, and hence, $\Theta \vdash P_2 \simeq^{\leqslant} Q$ by transitivity. Then $\Theta \vdash P_2 : e$.

Case 2. $e = (\widehat{\alpha}^+ :\geqslant Q)$. Then $\Theta \vdash P_1 \geqslant Q$, and hence, $\Theta \vdash P_2 \geqslant Q$ by transitivity. Then $\Theta \vdash P_2 : e$.

■

Corollary 28 (Constraint Satisfaction is stable under Equivalence)**.**
If $\Xi \vdash \widehat{\sigma}_1 : C$ *and* $\Xi \vdash \widehat{\sigma}_1 \simeq^{\leqslant} \widehat{\sigma}_2 : \mathtt{dom}\,(C)$ *then* $\Xi \vdash \widehat{\sigma}_2 : C$*;*
if $\Xi \vdash \widehat{\sigma}_1 : UC$ *and* $\Xi \vdash \widehat{\sigma}_1 \simeq^{\leqslant} \widehat{\sigma}_2 : \mathtt{dom}\,(C)$ *then* $\Xi \vdash \widehat{\sigma}_2 : UC$.

Corollary 29 (Normalization preserves Constraint Satisfaction)**.**
If $\Xi \vdash \widehat{\sigma} : C$ *then* $\Xi \vdash \mathtt{nf}\,(\widehat{\sigma}) : C$*;*
if $\Xi \vdash \widehat{\sigma} : UC$ *then* $\Xi \vdash \mathtt{nf}\,(\widehat{\sigma}) : UC$.

C.3.11 Positive Subtyping

Observation 13 (Positive Subtyping is Deterministic)**.** *For fixed* Θ, Ξ, P, *and* Q, *if* $\Theta;\ \Xi \vDash P \geqslant Q \dashv C$ *and* $\Theta;\ \Xi \vDash P \geqslant Q \dashv C'$ *then* $C = C'$.

Proof We prove it by induction on $\Theta;\ \Xi \vDash P \geqslant Q \dashv C$. First, it is easy to see that the rule applied to infer $\Theta;\ \Xi \vDash P \geqslant Q \dashv C$ uniquely depends on the input, and those, it is the same rule that is inferring $\Theta;\ \Xi \vDash P \geqslant Q \dashv C'$. Second, the premises of each rule are deterministic on the input: unification is deterministic by Observation 7, upgrade is deterministic by Observation 11, the choice of the fresh algorithmic variables is deterministic by convention, as discussed in Section A.2.2, positive subtyping by the induction hypothesis. ■

Lemma 86 (Soundness of the Positive Subtyping)**.** *If* $\Theta \vdash^{\supseteq} \Xi$, $\Theta \vdash Q$, $\Theta\,;\mathtt{dom}\,(\Xi) \vdash P$, *and* $\Theta;\ \Xi \vDash P \geqslant Q \dashv C$, *then* $\Xi \vdash C : \mathtt{fav} P$ *and for any normalized* $\widehat{\sigma}$ *such that* $\Xi \vdash \widehat{\sigma} : C$, $\Theta \vdash [\widehat{\sigma}]P \geqslant Q$.

Proof We prove it by induction on $\Theta;\ \Xi \vDash P \geqslant Q \dashv C$. Let us consider the last rule to infer this judgment.

Case 1. (UVar$^{\geqslant}$) then $\Theta;\ \Xi \vDash P \geqslant Q \dashv C$ has shape $\Theta;\ \Xi \vDash \widehat{\alpha}^+ \geqslant P' \dashv (\widehat{\alpha}^+ :\geqslant Q')$ where $\widehat{\alpha}^+\{\Theta_0\} \in \Xi$ and $\mathtt{upgrade}\ \Theta \vdash P' \ \mathtt{to}\ \Theta_0 = Q'$.

Notice that $\widehat{\alpha}^+\{\Theta_0\} \in \Xi$ and $\Theta \vdash^{\supseteq} \Xi$ implies $\Theta = \Theta_0, \overrightarrow{\alpha^{\pm}}$ for some $\overrightarrow{\alpha^{\pm}}$, hence, the soundness of upgrade (Lemma 103) is applicable:

1. $\Theta_0 \vdash Q'$ and
2. $\Theta \vdash Q' \geqslant P$.

Since $\widehat{\alpha}^+\{\Theta_0\} \in \Xi$ and $\Theta_0 \vdash Q'$, it is clear that $\Xi \vdash (\widehat{\alpha}^+ :\geqslant Q') : \widehat{\alpha}^+$.

It is left to show that $\Theta \vdash [\widehat{\sigma}]\widehat{\alpha}^+ \geqslant P'$ for any normalized $\widehat{\sigma}$ s.t. $\Xi \vdash \widehat{\sigma} : (\widehat{\alpha}^+ :\geqslant Q')$. The latter means that $\Xi(\widehat{\alpha}^+) \vdash [\widehat{\sigma}]\widehat{\alpha}^+ \geqslant Q'$, i.e. $\Theta_0 \vdash [\widehat{\sigma}]\widehat{\alpha}^+ \geqslant Q'$. By weakening the context to Θ and combining this judgment transitively with $\Theta \vdash Q' \geqslant P$, we have $\Theta \vdash [\widehat{\sigma}]\widehat{\alpha}^+ \geqslant P$, as required.

Case 2. ($\textsc{Var}^{\geqslant}_{+}$) then $\Theta;\ \Xi \vDash P \geqslant Q \dashv C$ has shape $\Theta;\ \Xi \vDash \alpha^+ \geqslant \alpha^+ \dashv \cdot$. Then $\mathtt{fav}\alpha^+ = \emptyset$, and $C = \cdot$ satisfies $\Xi \vdash C : \cdot$. Since $\mathtt{fav}\alpha^+ = \emptyset$, application of any substitution $\widehat{\sigma}$ does not change α^+, i.e. $[\widehat{\sigma}]\alpha^+ = \alpha^+$. Therefore, $\Theta \vdash [\widehat{\sigma}]\alpha^+ \geqslant \alpha^+$ holds by ($\textsc{Var}^{\leqslant}_{-}$).

Case 3. ($\downarrow^{\geqslant}$) then $\Theta;\ \Xi \vDash P \geqslant Q \dashv C$ has shape $\Theta;\ \Xi \vDash \downarrow N \geqslant \downarrow M \dashv C$.
Then the next step of the algorithm is the unification of $\mathtt{nf}\,(N)$ and $\mathtt{nf}\,(M)$, and it returns the resulting unification constraint $UC = C$ as the result. By the soundness of unification (Lemma 72), $\Xi \vdash C : \mathtt{fav}(N)$ and for any normalized $\widehat{\sigma}$, $\Xi \vdash \widehat{\sigma} : C$ implies $[\widehat{\sigma}]\mathtt{nf}\,(N) = \mathtt{nf}\,(M)$, then we rewrite the left-hand side by Lemma 43: $\mathtt{nf}\,([\widehat{\sigma}]N) = \mathtt{nf}\,(M)$ and apply Lemma 48: $\Theta \vdash [\widehat{\sigma}]N \simeq^{\leqslant} M$, then by ($\uparrow^{\leqslant}$), $\Theta \vdash \downarrow[\widehat{\sigma}]N \geqslant \downarrow M$.

Case 4. ($\exists^{\geqslant}$) then $\Theta;\ \Xi \vDash P \geqslant Q \dashv C$ has shape $\Theta;\ \Xi \vDash \exists\overrightarrow{\alpha^-}.\ P' \geqslant \exists\overrightarrow{\beta^-}.\ Q' \dashv C$ s.t. either $\overrightarrow{\alpha^-}$ or $\overrightarrow{\beta^-}$ is not empty.
Then the algorithm creates fresh unification variables $\overrightarrow{\widehat{\alpha}^-}\{\Theta, \overrightarrow{\beta^-}\}$, substitutes the old $\overrightarrow{\alpha^-}$ with them in P', and makes the recursive call: $\Theta, \overrightarrow{\beta^-};\ \Xi, \overrightarrow{\widehat{\alpha}^-}\{\Theta, \overrightarrow{\beta^-}\} \vDash [\overrightarrow{\widehat{\alpha}^-}/\overrightarrow{\alpha^-}]P' \geqslant Q' \dashv C'$, returning as the result $C = C' \setminus \overrightarrow{\widehat{\alpha}^-}$.

Let us take an arbitrary normalized $\widehat{\sigma}$ s.t. $\Xi \vdash \widehat{\sigma} : C' \setminus \overrightarrow{\widehat{\alpha}^-}$. We wish to show $\Theta \vdash [\widehat{\sigma}]P \geqslant Q$, i.e. $\Theta \vdash \exists\overrightarrow{\alpha^-}.\ [\widehat{\sigma}]P' \geqslant \exists\overrightarrow{\beta^-}.\ Q'$. To do that, we apply ($\exists^{\geqslant}$), and what is left to show is $\Theta, \overrightarrow{\beta^-} \vdash [\overrightarrow{N}/\overrightarrow{\alpha^-}][\widehat{\sigma}]P' \geqslant Q'$ for some $\overrightarrow{N}$. If we construct a normalized $\widehat{\sigma}'$ such that $\Xi, \overrightarrow{\widehat{\alpha}^-}\{\Theta, \overrightarrow{\beta^-}\} \vdash \widehat{\sigma}' : C'$ and for some $\overrightarrow{N}$, $[\overrightarrow{N}/\overrightarrow{\alpha^-}][\widehat{\sigma}]P' = [\widehat{\sigma}'][\overrightarrow{\widehat{\alpha}^-}/\overrightarrow{\alpha^-}]P'$, we can apply the induction hypothesis to $\Theta, \overrightarrow{\beta^-};\ \Xi, \overrightarrow{\widehat{\alpha}^-}\{\Theta, \overrightarrow{\beta^-}\} \vDash [\overrightarrow{\widehat{\alpha}^-}/\overrightarrow{\alpha^-}]P \geqslant Q \dashv C'$ and infer the required subtyping.

Let us construct such $\widehat{\sigma}'$ by extending $\widehat{\sigma}$ with $\overrightarrow{\widehat{\alpha}^-}$ mapped to the corresponding types in C':

$$[\widehat{\sigma}']\widehat{\beta}^{\pm} = \begin{cases} [\widehat{\sigma}]\widehat{\beta}^{\pm} & \text{if } \widehat{\beta}^{\pm} \in \mathtt{dom}\,(C') \setminus \overrightarrow{\widehat{\alpha}^-} \\ \mathtt{nf}\,(N) & \text{if } \widehat{\beta}^{\pm} \in \overrightarrow{\widehat{\alpha}^-} \text{ and } (\widehat{\beta}^{\pm} :\simeq N) \in SC' \end{cases}$$

It is easy to see that $\widehat{\sigma}'$ is normalized: it inherits this property from $\widehat{\sigma}$. Let us show that $\Xi, \overrightarrow{\widehat{\alpha}^-}\{\Theta, \overrightarrow{\beta^-}\} \vdash \widehat{\sigma}' : C'$. Let us take an arbitrary entry e from C' restricting a variable $\widehat{\beta}^{\pm}$. Suppose $\widehat{\beta}^{\pm} \in \mathtt{dom}\,(C') \setminus \overrightarrow{\widehat{\alpha}^-}$. Then $(\Xi, \overrightarrow{\widehat{\alpha}^-}\{\Theta, \overrightarrow{\beta^-}\})(\widehat{\beta}^{\pm}) \vdash [\widehat{\sigma}']\widehat{\beta}^{\pm} : e$ is rewritten as $\Xi(\widehat{\beta}^{\pm}) \vdash [\widehat{\sigma}]\widehat{\beta}^{\pm} : e$, which holds since $\Xi \vdash \widehat{\sigma} : C'$. Suppose $\widehat{\beta}^{\pm} = \widehat{\alpha}_i^{-} \in \overrightarrow{\widehat{\alpha}^-}$. Then $e = (\widehat{\alpha}_i^{-} :\simeq N)$ for some N, $[\widehat{\sigma}']\widehat{\alpha}_i^{-} = \mathtt{nf}\,(N)$ by the definition, and $\Theta, \overrightarrow{\beta^-} \vdash \mathtt{nf}\,(N) : (\widehat{\alpha}_i^{-} :\simeq N)$ by ($:\simeq^{\textsc{sat}}_{-}$), since $\Theta \vdash \mathtt{nf}\,(N) \simeq^{\leqslant} N$ by Lemma 48.

Finally, let us show that $[\overrightarrow{N}/\overrightarrow{\alpha^-}][\widehat{\sigma}]P' = [\widehat{\sigma}'][\overrightarrow{\widehat{\alpha}^-}/\overrightarrow{\alpha^-}]P'$. For N_i, we take the *normalized* type restricting $\widehat{\alpha}_i^{-}$ in C'. Let us take an arbitrary variable from P.

1. If this variable is a unification variable $\widehat{\beta}^{\pm}$, then $[\overrightarrow{N}/\overrightarrow{\alpha^-}][\widehat{\sigma}]\widehat{\beta}^{\pm} = [\widehat{\sigma}]\widehat{\beta}^{\pm}$, since $\Xi \vdash \widehat{\sigma} : C' \setminus \overrightarrow{\widehat{\alpha}^-}$ and $\mathtt{dom}\,(\Xi) \cap \overrightarrow{\alpha^-} = \emptyset$.

Notice that $\widehat{\beta}^{\pm} \in \mathtt{dom}\,(\Xi)$, which is disjoint from $\overrightarrow{\widehat{\alpha}^{-}}$, that is $\widehat{\beta}^{\pm} \in \mathtt{dom}\,(C') \setminus \overrightarrow{\widehat{\alpha}^{-}}$. This way, $[\widehat{\sigma}'][\overrightarrow{\widehat{\alpha}^{-}}/\overrightarrow{\alpha^{-}}]\widehat{\beta}^{\pm} = [\widehat{\sigma}']\widehat{\beta}^{\pm} = [\widehat{\sigma}]\widehat{\beta}^{\pm}$ by the definition of $\widehat{\sigma}'$,

2. If this variable is a regular variable $\beta^{\pm} \notin \overrightarrow{\alpha^{-}}$, then $[\overrightarrow{N}/\overrightarrow{\alpha^{-}}][\widehat{\sigma}]\beta^{\pm} = \beta^{\pm}$ and $[\widehat{\sigma}'][\overrightarrow{\widehat{\alpha}^{-}}/\overrightarrow{\alpha^{-}}]\beta^{\pm} = \beta^{\pm}$.

3. If this variable is a regular variable $\alpha_i^{-} \in \overrightarrow{\alpha^{-}}$, then $[\overrightarrow{N}/\overrightarrow{\alpha^{-}}][\widehat{\sigma}]\alpha_i^{-} = N_i = \mathtt{nf}\,(N_i)$ (the latter equality holds since N_i is normalized) and $[\widehat{\sigma}'][\overrightarrow{\widehat{\alpha}^{-}}/\overrightarrow{\alpha^{-}}]\alpha_i^{-} = [\widehat{\sigma}']\widehat{\alpha}_i^{-} = \mathtt{nf}\,(N_i)$.

■

Lemma 87 (Completeness of the Positive Subtyping)**.** *Suppose that* $\Theta \vdash^{\supseteq} \Xi$, $\Theta \vdash Q$ *and* $\Theta\,;\mathtt{dom}\,(\Xi) \vdash P$. *Then for any* $\Xi \vdash \widehat{\sigma} : \mathtt{fav}(P)$ *such that* $\Theta \vdash [\widehat{\sigma}]P \geqslant Q$, *there exists* Θ; $\Xi \vDash P \geqslant Q \dashv C$ *and moreover,* $\Xi \vdash \widehat{\sigma} : C$.

Proof Let us prove this lemma by induction on $\Theta \vdash [\widehat{\sigma}]P \geqslant Q$. Let us consider the last rule used in the derivation, but first, consider the base case for the substitution $[\widehat{\sigma}]P$:

Case 1. $P = \exists\overrightarrow{\beta^{-}}.\,\widehat{\alpha}^{+}$ (for potentially empty $\overrightarrow{\beta^{-}}$)

Then by assumption, $\Theta \vdash \exists\overrightarrow{\beta^{-}}.\,[\widehat{\sigma}]\widehat{\alpha}^{+} \geqslant Q$ (where $\overrightarrow{\beta^{-}} \cap \mathtt{fv}\,[\widehat{\sigma}]\widehat{\alpha}^{+} = \emptyset$). Let us decompose Q as $Q = \exists\overrightarrow{\gamma^{-}}.\,Q_0$, where Q_0 does not start with $\exists$.

By inversion, $\Theta\,;\mathtt{dom}\,(\Xi) \vdash \exists\overrightarrow{\beta^{-}}.\,\widehat{\alpha}^{+}$ implies $\widehat{\alpha}^{+}\{\Theta_0\} \in \Xi$ for some $\Theta_0 \subseteq \Theta$.

By Lemma 18 applied twice, $\Theta \vdash \exists\overrightarrow{\beta^{-}}.\,[\widehat{\sigma}]\widehat{\alpha}^{+} \geqslant \exists\overrightarrow{\gamma^{-}}.\,Q_0$ implies $\Theta, \overrightarrow{\gamma^{-}} \vdash [\overrightarrow{N}/\overrightarrow{\beta^{-}}][\widehat{\sigma}]\widehat{\alpha}^{+} \geqslant Q_0$ for some N, and since $\overrightarrow{\beta^{-}} \cap \mathtt{fv}\,([\widehat{\sigma}]\widehat{\alpha}^{+}) \subseteq \overrightarrow{\beta^{-}} \cap \Xi(\widehat{\alpha}^{+}) \subseteq \overrightarrow{\beta^{-}} \cap \Theta = \cdot$, $[\overrightarrow{N}/\overrightarrow{\beta^{-}}][\widehat{\sigma}]\widehat{\alpha}^{+} = [\widehat{\sigma}]\widehat{\alpha}^{+}$, that is $\Theta, \overrightarrow{\gamma^{-}} \vdash [\widehat{\sigma}]\widehat{\alpha}^{+} \geqslant Q_0$.

When algorithm tires to infer the subtyping Θ; $\Xi \vDash \exists\overrightarrow{\beta^{-}}.\,\widehat{\alpha}^{+} \geqslant \exists\overrightarrow{\gamma^{-}}.\,Q_0 \dashv C$, it applies ($\exists^{\geqslant}$), which reduces the problem to $\Theta, \overrightarrow{\gamma^{-}}$; $\Xi, \overrightarrow{\widehat{\beta}^{-}}\{\Theta, \overrightarrow{\gamma^{-}}\} \vDash [\overrightarrow{\widehat{\beta}^{-}}/\overrightarrow{\beta^{-}}]\widehat{\alpha}^{+} \geqslant Q_0 \dashv C$, which is equivalent to $\Theta, \overrightarrow{\gamma^{-}}$; $\Xi, \overrightarrow{\widehat{\beta}^{-}}\{\Theta, \overrightarrow{\gamma^{-}}\} \vDash \widehat{\alpha}^{+} \geqslant Q_0 \dashv C$.

Next, the algorithm tries to apply (UVar$^{\geqslant}$) and the resulting restriction is $C = (\widehat{\alpha}^{+} :\geqslant Q_0')$ where $\mathtt{upgrade}\,\Theta, \overrightarrow{\gamma^{-}} \vdash Q_0\ \mathtt{to}\ \Theta_0 = Q_0'$.

Why does the upgrade procedure terminate? Because $[\widehat{\sigma}]\widehat{\alpha}^{+}$ satisfies the pre-conditions of the completeness of the upgrade (Lemma 83):

1. $\Theta_0 \vdash [\widehat{\sigma}]\widehat{\alpha}^{+}$ because $\Xi \vdash \widehat{\sigma} : \widehat{\alpha}^{+}$ and $\widehat{\alpha}^{+}\{\Theta_0\} \in \Xi$,
2. $\Theta, \overrightarrow{\gamma^{-}} \vdash [\widehat{\sigma}]\widehat{\alpha}^{+} \geqslant Q_0$ as noted above

Moreover, the completeness of upgrade also says that Q_0' is *the least* supertype of Q_0 among types well-formed in Θ_0, that is $\Theta_0 \vdash [\widehat{\sigma}]\widehat{\alpha}^{+} \geqslant Q_0'$, which means $\Xi \vdash \widehat{\sigma} : (\widehat{\alpha}^{+} :\geqslant Q_0')$, that is $\Xi \vdash \widehat{\sigma} : C$.

Case 2. $\Theta \vdash [\widehat{\sigma}]P \geqslant Q$ is derived by (Var$_{+}^{\geqslant}$)

Then $P = [\widehat{\sigma}]P = \alpha^{+} = Q$, where the first equality holds because P is not a unification variable: it has been covered by case 1; and the second equality hold because (Var$_{+}^{\geqslant}$) was applied.

The algorithm applies ($\textsc{Var}_+^{\geqslant}$) and infers $C = \cdot$, i.e. $\Theta;\ \Xi \vDash \alpha^+ \geqslant \alpha^+ \dashv \cdot$. Then $\Xi \vdash \widehat{\sigma} : \cdot$ holds trivially.

Case 3. $\Theta \vdash [\widehat{\sigma}]P \geqslant Q$ is derived by ($\downarrow^{\geqslant}$),

Then $P = \downarrow N$, since the substitution $[\widehat{\sigma}]P$ must preserve the top-level constructor of $P \neq \widehat{\alpha}^+$ (the case $P = \widehat{\alpha}^+$ has been covered by case 1), and $Q = \downarrow M$, and by inversion, $\Theta \vdash [\widehat{\sigma}]N \simeq^{\leqslant} M$.

Since both types start with $\downarrow$, the algorithm tries to apply ($\downarrow^{\geqslant}$): $\Theta;\ \Xi \vDash \downarrow N \geqslant \downarrow M \dashv C$. The premise of this rule is the unification of $\mathtt{nf}\,(N)$ and $\mathtt{nf}\,(M)$: $\Theta; \Xi \vDash \mathtt{nf}\,(N) \overset{u}{\simeq} \mathtt{nf}\,(M) \dashv UC$. And the algorithm returns it as a subtyping constraint $C = UC$.

To demonstrate that the unification terminates ant $\widehat{\sigma}$ satisfies the resulting constraints, we apply the completeness of the unification algorithm (Lemma 73). In order to do that, we need to provide a substitution unifying $\mathtt{nf}\,(N)$ and $\mathtt{nf}\,(M)$. Let us show that $\mathtt{nf}\,(\widehat{\sigma})$ is such a substitution.

- $\mathtt{nf}\,(N)$ and $\mathtt{nf}\,(M)$ are normalized
- $\Theta\,;\mathtt{dom}\,(\Xi) \vdash \mathtt{nf}\,(N)$ because $\Theta\,;\mathtt{dom}\,(\Xi) \vdash N$ (Corollary 24)
- $\Theta \vdash \mathtt{nf}\,(M)$ because $\Theta \vdash M$ (Corollary 14)
- $\Xi \vdash \mathtt{nf}\,(\widehat{\sigma}) : \mathtt{fav}(P)$ because $\Xi \vdash \widehat{\sigma} : \mathtt{fav}(P)$ (Corollary 25)
-

$$\begin{aligned}
\Theta \vdash [\widehat{\sigma}]N \simeq^{\leqslant} M &\Rightarrow [\widehat{\sigma}]N \simeq^{D} M && \text{by Lemma 34}\\
&\Rightarrow \mathtt{nf}\,([\widehat{\sigma}]N) = \mathtt{nf}\,(M) && \text{by Lemma 44}\\
&\Rightarrow [\mathtt{nf}\,(\widehat{\sigma})]\mathtt{nf}\,(N) = \mathtt{nf}\,(M) && \text{by Lemma 43}
\end{aligned}$$

By the completeness of the unification, $\Theta; \Xi \vDash N \overset{u}{\simeq} M \dashv UC$ exists, and $\Xi \vdash \mathtt{nf}\,(\widehat{\sigma}) : UC$, and by Corollary 28, $\Xi \vdash \widehat{\sigma} : UC$.

Case 4. $\Theta \vdash [\widehat{\sigma}]P \geqslant Q$ is derived by ($\exists^{\geqslant}$).

We should only consider the case when the substitution $[\widehat{\sigma}]P$ results in the existential type $\exists\overrightarrow{\alpha}.\ P''$ (for $P'' \neq \exists \ldots$) by congruence, i.e. $P = \exists\overrightarrow{\alpha}.\ P'$ (for $P' \neq \exists \ldots$) and $[\widehat{\sigma}]P' = P''$. This is because the case when $P = \exists\overrightarrow{\beta}.\ \widehat{\alpha}^+$ has been covered (case 1), and thus, the substitution $\widehat{\sigma}$ must preserve all the outer quantifiers of P and does not generate any new ones.

This way, $P = \exists\overrightarrow{\alpha}.\ P'$, $[\widehat{\sigma}]P = \exists\overrightarrow{\alpha}.\ [\widehat{\sigma}]P'$ (assuming $\overrightarrow{\alpha}$ does not intersect with the range of $\widehat{\sigma}$) and $Q = \exists\overrightarrow{\beta}.\ Q'$, where either $\overrightarrow{\alpha}$ or $\overrightarrow{\beta}$ is not empty.

By inversion, $\Theta \vdash [\sigma][\widehat{\sigma}]P' \geqslant Q'$ for some $\Theta, \overrightarrow{\beta} \vdash \sigma : \overrightarrow{\alpha}$. Since σ and $\widehat{\sigma}$ have disjoint domains, and the range of one does not intersect with the domain of the other, they commute, i.e. $\Theta, \overrightarrow{\beta} \vdash [\widehat{\sigma}][\sigma]P' \geqslant Q'$ (notice that the tree inferring this judgement is a proper subtree of the tree inferring $\Theta \vdash [\widehat{\sigma}]P \geqslant Q$).

At the next step, the algorithm creates fresh (disjoint with $\mathtt{fav}\,P'$) unification variables $\widehat{\overrightarrow{\alpha^{-}}}$, replaces $\overrightarrow{\alpha^{-}}$ with them in P', and makes the recursive call: $\Theta, \overrightarrow{\beta^{-}};\ \Xi, \widehat{\overrightarrow{\alpha^{-}}}\{\Theta, \overrightarrow{\beta^{-}}\} \vDash P_0 \geqslant Q' \dashv C_1$, (where $P_0 = [\widehat{\overrightarrow{\alpha^{-}}}/\overrightarrow{\alpha^{-}}]P'$), returning $C_1 \setminus \widehat{\overrightarrow{\alpha^{-}}}$ as the result.

To show that the recursive call terminates and that $\Xi \vdash \widehat{\sigma} : C_1 \setminus \widehat{\overrightarrow{\alpha^{-}}}$, it suffices to build $\Xi, \widehat{\overrightarrow{\alpha^{-}}}\{\Theta, \overrightarrow{\beta^{-}}\} \vdash \widehat{\sigma}_0 : \mathtt{fav}(P_0)$—an extension of $\widehat{\sigma}$ with $\widehat{\overrightarrow{\alpha^{-}}} \cap \mathtt{fav}(P_0)$ such that $\Theta, \overrightarrow{\beta^{-}} \vdash [\widehat{\sigma}_0]P_0 \geqslant Q$. Then by the induction hypothesis, $\Xi, \widehat{\overrightarrow{\alpha^{-}}}\{\Theta, \overrightarrow{\beta^{-}}\} \vdash \widehat{\sigma}_0 : C_1$, and hence, $\Xi \vdash \widehat{\sigma} : C_1 \setminus \widehat{\overrightarrow{\alpha^{-}}}$, as required.

Let us construct such a substitution $\widehat{\sigma}_0$:

$$[\widehat{\sigma}_0]\widehat{\beta^{\pm}} = \begin{cases} [\sigma]\alpha_i^{-} & \text{if } \widehat{\beta^{\pm}} = \widehat{\alpha_i}^{-} \in \widehat{\overrightarrow{\alpha^{-}}} \cap \mathtt{fav}(P_0) \\ [\widehat{\sigma}]\widehat{\beta^{\pm}} & \text{if } \widehat{\beta^{\pm}} \in \mathtt{fav}(P') \end{cases}$$

It is easy to see $\Xi, \widehat{\overrightarrow{\alpha^{-}}}\{\Theta, \overrightarrow{\beta^{-}}\} \vdash \widehat{\sigma}_0 : \mathtt{fav}(P_0)$: $\mathtt{fav}(P_0) = \mathtt{fav}([\widehat{\overrightarrow{\alpha^{-}}}/\overrightarrow{\alpha^{-}}]P') = \widehat{\overrightarrow{\alpha^{-}}} \cap \mathtt{fav}(P_0) \cup \mathtt{fav}(P')$. Then

1. for $\widehat{\alpha_i}^{-} \in \widehat{\overrightarrow{\alpha^{-}}} \cap \mathtt{fav}(P_0)$, $(\Xi, \widehat{\overrightarrow{\alpha^{-}}}\{\Theta, \overrightarrow{\beta^{-}}\})(\widehat{\alpha_i}^{-}) \vdash [\widehat{\sigma}_0]\widehat{\alpha_i}^{-}$, i.e. $\Theta, \overrightarrow{\beta^{-}} \vdash [\sigma]\alpha_i^{-}$ holds since $\Theta, \overrightarrow{\beta^{-}} \vdash \sigma : \overrightarrow{\alpha^{-}}$,
2. for $\widehat{\beta^{\pm}} \in \mathtt{fav}(P') \subseteq \mathtt{dom}(\Xi)$, $(\Xi, \widehat{\overrightarrow{\alpha^{-}}}\{\Theta, \overrightarrow{\beta^{-}}\})(\widehat{\beta^{\pm}}) \vdash [\widehat{\sigma}_0]\widehat{\beta^{\pm}}$, i.e. $\Xi(\widehat{\beta^{\pm}}) \vdash [\widehat{\sigma}]\widehat{\beta^{\pm}}$ holds since $\Xi \vdash \widehat{\sigma} : \mathtt{fav}(P)$ and $\widehat{\beta^{\pm}} \in \mathtt{fav}(P') = \mathtt{fav}(P)$.

Now, let us show that $\Theta, \overrightarrow{\beta^{-}} \vdash [\widehat{\sigma}_0]P_0 \geqslant Q$. To do that, we notice that $[\widehat{\sigma}_0]P_0 = [\widehat{\sigma}][\sigma][\overrightarrow{\alpha^{-}}/\widehat{\overrightarrow{\alpha^{-}}}]P_0$: let us consider an arbitrary variable appearing freely in P_0:

1. if this variable is a algorithmic variable $\widehat{\alpha_i}^{-} \in \widehat{\overrightarrow{\alpha^{-}}}$, then $[\widehat{\sigma}_0]\widehat{\alpha_i}^{-} = [\sigma]\alpha_i^{-}$ and $[\widehat{\sigma}][\sigma][\overrightarrow{\alpha^{-}}/\widehat{\overrightarrow{\alpha^{-}}}]\widehat{\alpha_i}^{-} = [\widehat{\sigma}][\sigma]\alpha_i^{-} = [\sigma]\alpha_i^{-}$,
2. if this variable is a algorithmic variable $\widehat{\beta^{\pm}} \in \mathtt{fav}(P_0) \setminus \widehat{\overrightarrow{\alpha^{-}}} = \mathtt{fav}(P')$, then $[\widehat{\sigma}_0]\widehat{\beta^{\pm}} = [\widehat{\sigma}]\widehat{\beta^{\pm}}$ and $[\widehat{\sigma}][\sigma][\overrightarrow{\alpha^{-}}/\widehat{\overrightarrow{\alpha^{-}}}]\widehat{\beta^{\pm}} = [\widehat{\sigma}][\sigma]\widehat{\beta^{\pm}} = [\widehat{\sigma}]\widehat{\beta^{\pm}}$,
3. if this variable is a regular variable from $\mathtt{fv}(P_0)$, both substitutions do not change it: $\widehat{\sigma}_0$, $\widehat{\sigma}$ and $\overrightarrow{\alpha^{-}}/\widehat{\overrightarrow{\alpha^{-}}}$ act on algorithmic variables, and σ is defined on $\overrightarrow{\alpha^{-}}$, however, $\overrightarrow{\alpha^{-}} \cap \mathtt{fv}(P_0) = \emptyset$.

This way, $[\widehat{\sigma}_0]P_0 = [\widehat{\sigma}][\sigma][\overrightarrow{\alpha^{-}}/\widehat{\overrightarrow{\alpha^{-}}}]P_0 = [\widehat{\sigma}][\sigma]P'$, and thus, $\Theta, \overrightarrow{\beta^{-}} \vdash [\widehat{\sigma}_0]P_0 \geqslant Q'$.

■

C.3.12 Subtyping Constraint Merge

Observation 14 (Constraint Entry Merge is Deterministic)**.** *For fixed Θ, e_1, e_2, if $\Theta \vdash e_1 \,\&\, e_2 = e$ and $\Theta \vdash e_1 \,\&\, e_2 = e'$ then $e = e'$.*

Proof First, notice that the shape of e_1 and e_2 uniquely determines the rule applied to infer $\Theta \vdash e_1 \,\&\, e_2 = e$, which is consequently, the same rule used to infer $\Theta \vdash e_1 \,\&\, e_2 = e'$. Second, notice that the premises of each rule are deterministic on the input: the positive subtyping is deterministic by Observation 13, and the least upper bound is deterministic by Observation 10. ■

Observation 15 (Subtyping Constraint Merge is Deterministic)**.** *Suppose that* $\Xi \vdash C_1$ *and* $\Xi \vdash C_2$ *If* $\Xi \vdash C_1 \,\&\, C_2 = C$ *and* $\Xi \vdash C_1 \,\&\, C_2 = C'$ *are defined then* $C = C'$.

Proof The proof is analogous to the proof of Observation 6 but uses Observation 14 to show that the merge of the matching constraint entries is fixed. ■

Lemma 88 (Soundness of Constraint Entry Merge)**.** *For a fixed context* Θ*, suppose that* $\Theta \vdash e_1$ *and* $\Theta \vdash e_2$*. If* $\Theta \vdash e_1 \,\&\, e_2 = e$ *is defined then*

1. $\Theta \vdash e$
2. *For any* $\Theta \vdash P$, $\Theta \vdash P : e$ *implies* $\Theta \vdash P : e_1$ *and* $\Theta \vdash P : e_2$

Proof Let us consider the rule forming $\Theta \vdash e_1 \,\&\, e_2 = e$.

Case 1. $(\simeq \,\&^+\, \simeq)$, i.e. $\Theta \vdash e_1 \,\&\, e_2 = e$ has form $\Theta \vdash (\widehat{\alpha}^+ :\simeq Q) \,\&\, (\widehat{\alpha}^+ :\simeq Q') = (\widehat{\alpha}^+ :\simeq Q)$ and $\mathtt{nf}\,(Q) = \mathtt{nf}\,(Q')$. The latter implies $\Theta \vdash Q \simeq^{\leqslant} Q'$ by Lemma 48. Then

1. $\Theta \vdash e$, i.e. $\Theta \vdash \widehat{\alpha}^+ :\simeq Q$ holds by assumption;
2. by inversion, $\Theta \vdash P : (\widehat{\alpha}^+ :\simeq Q)$ means $\Theta \vdash P \simeq^{\leqslant} Q$, and by transitivity of equivalence (Corollary 10), $\Theta \vdash P \simeq^{\leqslant} Q'$. Thus, $\Theta \vdash P : e_1$ and $\Theta \vdash P : e_2$ hold by $(:\simeq^{\mathrm{SAT}}_{+})$.

Case 2. $(\simeq \,\&^-\, \simeq)$ the negative case is proved in exactly the same way as the positive one.

Case 3. $(\geqslant \,\&^+\, \geqslant)$ Then e_1 is $\widehat{\alpha}^+ :\geqslant Q_1$, e_2 is $\widehat{\alpha}^+ :\geqslant Q_2$, and $e_1 \,\&\, e_2 = e$ is $\widehat{\alpha}^+ :\geqslant Q$ where Q is the least upper bound of Q_1 and Q_2. Then by Lemma 80,

- $\Theta \vdash Q$,
- $\Theta \vdash Q \geqslant Q_1$,
- $\Theta \vdash Q \geqslant Q_2$.

Let us show the required properties.

- $\Theta \vdash e$ holds from $\Theta \vdash Q$,
- Assuming $\Theta \vdash P : e$, by inversion, we have $\Theta \vdash P \geqslant Q$. Combining it transitively with $\Theta \vdash Q \geqslant Q_1$, we have $\Theta \vdash P \geqslant Q_1$. Analogously, $\Theta \vdash P \geqslant Q_2$. Then $\Theta \vdash P : e_1$ and $\Theta \vdash P : e_2$ hold by $(:\geqslant^{\mathrm{SAT}}_{+})$.

Case 4. $(\geqslant \,\&^+\, \simeq)$ Then e_1 is $\widehat{\alpha}^+ :\geqslant Q_1$, e_2 is $\widehat{\alpha}^+ :\simeq Q_2$, where $\Theta;\ \cdot \vDash Q_2 \geqslant Q_1 \dashv \cdot$, and the resulting $e_1 \,\&\, e_2 = e$ is equal to e_2, that is $\widehat{\alpha}^+ :\simeq Q_2$.

Let us show the required properties.

- By assumption, $\Theta \vdash Q$, and hence $\Theta \vdash e$.
- Since $\mathtt{fav}(Q_2) = \emptyset$, $\Theta;\ \cdot \vDash Q_2 \geqslant Q_1 \dashv \cdot$ implies $\Theta \vdash Q_2 \geqslant Q_1$ by the soundness of positive subtyping (Lemma 86). Then let us take an arbitrary $\Theta \vdash P$ such that $\Theta \vdash P : e$. Since $e_2 = e$, $\Theta \vdash P : e_2$ holds immediately.

By inversion, $\Theta \vdash P : (\widehat{\alpha}^{+} :\simeq Q_2)$ means $\Theta \vdash P \simeq^{\leqslant} Q_2$, and then by transitivity of subtyping (Lemma 24), $\Theta \vdash P \geqslant Q_1$. Then $\Theta \vdash P : e_1$ holds by $(:\geqslant^{\text{SAT}}_{+})$.

Case 5. $(\simeq \&^{+} \geqslant)$ Thee proof is analogous to the previous case.

■

Lemma 89 (Soundness of Constraint Merge)**.** *Suppose that* $\Xi \vdash C_1 : \widehat{\Theta}_1$ *and* $\Xi \vdash C_2 : \widehat{\Theta}_2$ *and* $\Xi \vdash C_1 \,\&\, C_2 = C$ *is defined. Then*

1. $\Xi \vdash C : \widehat{\Theta}_1 \cup \widehat{\Theta}_2$,
2. *for any substitution* $\Xi \vdash \widehat{\sigma} : \widehat{\Theta}_1 \cup \widehat{\Theta}_2$, $\Xi \vdash \widehat{\sigma} : C$ *implies* $\Xi \vdash \widehat{\sigma} : C_1$ *and* $\Xi \vdash \widehat{\sigma} : C_2$.

Proof By definition, $\Xi \vdash C_1 \,\&\, C_2 = C$ consists of three parts: entries of C_1 that do not have matching entries of C_2, entries of C_2 that do not have matching entries of C_1, and the merge of matching entries.

Notice that $\widehat{\alpha}^{\pm} \in \widehat{\Theta}_1 \setminus \widehat{\Theta}_2$ if and only if there is an entry e in C_1 restricting $\widehat{\alpha}^{\pm}$, but there is no such entry in C_2. Therefore, for any $\widehat{\alpha}^{\pm} \in \widehat{\Theta}_1 \setminus \widehat{\Theta}_2$, there is an entry e in C restricting $\widehat{\alpha}^{\pm}$. Notice that $\Xi(\widehat{\alpha}^{\pm}) \vdash e$ holds since $\Xi \vdash C_1 : \widehat{\Theta}_1$.

Analogously, for any $\widehat{\beta}^{\pm} \in \widehat{\Theta}_2 \setminus \widehat{\Theta}_1$, there is an entry e in C restricting $\widehat{\beta}^{\pm}$. Notice that $\Xi(\widehat{\beta}^{\pm}) \vdash e$ holds since $\Xi \vdash C_2 : \widehat{\Theta}_2$.

Finally, for any $\widehat{\gamma}^{\pm} \in \widehat{\Theta}_1 \cap \widehat{\Theta}_2$, there is an entry e_1 in C_1 restricting $\widehat{\gamma}^{\pm}$ and an entry e_2 in C_2 restricting $\widehat{\gamma}^{\pm}$. Since $\Xi \vdash C_1 \,\&\, C_2 = C$ is defined, $\Xi(\widehat{\gamma}^{\pm}) \vdash e_1 \,\&\, e_2 = e$ restricting $\widehat{\gamma}^{\pm}$ is defined and belongs to C, moreover, $\Xi(\widehat{\gamma}^{\pm}) \vdash e$ by Lemma 88. This way, $\Xi \vdash C : \widehat{\Theta}_1 \cup \widehat{\Theta}_2$.

Let us show the second property. We take an arbitrary $\widehat{\sigma}$ such that $\Xi \vdash \widehat{\sigma} : \widehat{\Theta}_1 \cup \widehat{\Theta}_2$ and $\Xi \vdash \widehat{\sigma} : C$. To prove $\Xi \vdash \widehat{\sigma} : C_1$, we need to show that for any $e_1 \in C_1$, restricting $\widehat{\alpha}^{\pm}$, $\Xi(\widehat{\alpha}^{\pm}) \vdash [\widehat{\sigma}]\widehat{\alpha}^{\pm} : e_1$ holds.

Let us assume that $\widehat{\alpha}^{\pm} \notin \mathtt{dom}\,(C_2)$. It means that $C \ni e_1$, and then since $\Xi \vdash \widehat{\sigma} : C$, $\Xi(\widehat{\alpha}^{\pm}) \vdash [\widehat{\sigma}]\widehat{\alpha}^{\pm} : e_1$.

Otherwise, C_2 contains an entry e_2 restricting $\widehat{\alpha}^{\pm}$, and $C \ni e$ where $\Xi(\widehat{\alpha}^{\pm}) \vdash e_1 \,\&\, e_2 = e$. Then since $\Xi \vdash \widehat{\sigma} : C$, $\Xi(\widehat{\alpha}^{\pm}) \vdash [\widehat{\sigma}]\widehat{\alpha}^{\pm} : e$, and by Lemma 88, $\Xi(\widehat{\alpha}^{\pm}) \vdash [\widehat{\sigma}]\widehat{\alpha}^{\pm} : e_1$.

The proof of $\Xi \vdash \widehat{\sigma} : C_2$ is symmetric. ■

Lemma 90 (Completeness of Constraint Entry Merge)**.** *For a fixed context* Θ*, suppose that* $\Theta \vdash e_1$ *and* $\Theta \vdash e_2$ *are matching constraint entries.*

- *for a type* P *such that* $\Theta \vdash P : e_1$ *and* $\Theta \vdash P : e_2$, $\Theta \vdash e_1 \,\&\, e_2 = e$ *is defined and* $\Theta \vdash P : e$.
- *for a type* N *such that* $\Theta \vdash N : e_1$ *and* $\Theta \vdash N : e_2$, $\Theta \vdash e_1 \,\&\, e_2 = e$ *is defined and* $\Theta \vdash N : e$.

Proof Let us consider the shape of e_1 and e_2.

Case 1. e_1 is $\widehat{\alpha}^{+} :\simeq Q_1$ and e_2 is $\widehat{\alpha}^{+} :\simeq Q_2$. The proof repeats the corresponding case of Lemma 70

Case 2. e_1 is $\widehat{\alpha}^+ :\simeq Q_1$ and e_2 is $\widehat{\alpha}^+ :\geqslant Q_2$. Then $\Theta \vdash P : e_1$ means $\Theta \vdash P \simeq^{\leqslant} Q_1$, and $\Theta \vdash P : e_2$ means $\Theta \vdash P \geqslant Q_2$. Then by transitivity of subtyping, $\Theta \vdash Q_1 \geqslant Q_2$, which means $\Theta;\ \cdot \vDash Q_1 \geqslant Q_2 \dashv \cdot$ by Lemma 87. This way, $(\simeq \&^+ \geqslant)$ applies to infer $\Theta \vdash e_1 \,\&\, e_2 = e_1$, and $\Theta \vdash P : e_1$ holds by assumption.

Case 3. e_1 is $\widehat{\alpha}^+ :\geqslant Q_1$ and e_2 is $\widehat{\alpha}^+ :\geqslant Q_2$. Then $\Theta \vdash P : e_1$ means $\Theta \vdash P \geqslant Q_1$, and $\Theta \vdash P : e_2$ means $\Theta \vdash P \geqslant Q_2$. By the completeness of the least upper bound (Lemma 81), $\Theta \vDash Q_1 \vee Q_2 = Q$, and $\Theta \vdash P \geqslant Q$. This way, $(\geqslant \&^+ \geqslant)$ applies to infer $\Theta \vdash e_1 \,\&\, e_2 = (\widehat{\alpha}^+ :\geqslant Q)$, and $\Theta \vdash P : (\widehat{\alpha}^+ :\geqslant Q)$ holds by $(:\geqslant^{\mathrm{SAT}}_{+})$.

Case 4. The negative cases are proved symmetrically.

■

Lemma 91 (Completeness of Constraint Merge)**.** *Suppose that* $\Xi \vdash C_1 : \widehat{\Theta}_1$ *and* $\Xi \vdash C_2 : \widehat{\Theta}_2$. *If there exists a substitution* $\Xi \vdash \widehat{\sigma} : \widehat{\Theta}_1 \cup \widehat{\Theta}_2$ *such that* $\Xi \vdash \widehat{\sigma} : C_1$ *and* $\Xi \vdash \widehat{\sigma} : C_2$ *then* $\Xi \vdash C_1 \,\&\, C_2 = C$ *is defined.*

Proof By definition, $C_1 \,\&\, C_2$ is a union of

1. entries of C_1, which do not have matching entries in C_2,
2. entries of C_2, which do not have matching entries in C_1, and
3. the merge of matching entries.

This way, to show that $\Xi \vdash C_1 \,\&\, C_2 = C$ is defined, we need to demonstrate that each of these components is defined and satisfies the required property (that the result of $\widehat{\sigma}$ satisfies the corresponding constraint entry).

It is clear that the first two components of this union exist. Moreover, if e is an entry of C_i restricting $\widehat{\alpha}^{\pm} \notin \mathtt{dom}\,(C_2)$, then $\Xi \vdash \widehat{\sigma} : C_i$ implies $\Xi(\widehat{\alpha}^{\pm}) \vdash [\widehat{\sigma}]\widehat{\alpha}^{\pm} : e$,

Let us show that the third component exists. Let us take two entries $e_1 \in C_1$ and $e_2 \in C_2$ restricting the same variable $\widehat{\alpha}^{\pm}$. $\Xi \vdash \widehat{\sigma} : C_1$ means that $\Xi(\widehat{\alpha}^{\pm}) \vdash [\widehat{\sigma}]\widehat{\alpha}^{\pm} : e_1$ and $\Xi \vdash \widehat{\sigma} : C_2$ means $\Xi(\widehat{\alpha}^{\pm}) \vdash [\widehat{\sigma}]\widehat{\alpha}^{\pm} : e_2$. Then by Lemma 90, $\Xi(\widehat{\alpha}^{\pm}) \vdash e_1 \,\&\, e_2 = e$ is defined and $\Xi(\widehat{\alpha}^{\pm}) \vdash [\widehat{\sigma}]\widehat{\alpha}^{\pm} : e$.

■

C.3.13 Negative Subtyping

Observation 16 (Negative Algorithmic Subtyping is Deterministic)**.** *For fixed* Θ, Ξ, M, *and* N, *if* $\Theta;\ \Xi \vDash N \leqslant M \dashv C$ *and* $\Theta;\ \Xi \vDash N \leqslant M \dashv C'$ *then* $C = C'$.

Proof First, notice that the shape of the input uniquely determines the rule applied to infer $\Theta;\ \Xi \vDash N \leqslant M \dashv C$, which is consequently, the same rule used to infer $\Theta;\ \Xi \vDash N \leqslant M \dashv C'$.

Second, notice that for each of the inference rules, the premises are deterministic on the input. Specifically,

- $(\uparrow^{\leqslant})$ relies on unification, which is deterministic by Observation 7;

- $(\forall^{\leqslant})$ relies on the choice of fresh algorithmic variables, which is deterministic as discussed in Section A.2.2, and on the negative subtyping, which is deterministic by the induction hypothesis;
- $(\to^{\leqslant})$ uses the negative subtyping (deterministic by the induction hypothesis), the positive subtyping (Observation 13), and the merge of subtyping constraints (Observation 15);

■

Lemma 92 (Soundness of Negative Subtyping). *If* $\Theta \vdash^{\supseteq} \Xi$, $\Theta \vdash M$, $\Theta\,;\mathtt{dom}\,(\Xi) \vdash N$ *and* $\Theta;\ \Xi \vDash N \leqslant M \dashv C$, *then* $\Xi \vdash C : \mathtt{fav}(N)$ *and for any normalized* $\widehat{\sigma}$ *such that* $\Xi \vdash \widehat{\sigma} : C$, $\Theta \vdash [\widehat{\sigma}]N \leqslant M$.

Proof We prove it by induction on $\Theta;\ \Xi \vDash N \leqslant M \dashv C$.

Suppose that $\widehat{\sigma}$ is normalized and $\Xi \vdash \widehat{\sigma} : C$, Let us consider the last rule to infer this judgment.

Case 1. $(\to^{\leqslant})$. Then $\Theta;\ \Xi \vDash N \leqslant M \dashv C$ has shape $\Theta;\ \Xi \vDash P \to N' \leqslant Q \to M' \dashv C$
On the next step, the the algorithm makes two recursive calls: $\Theta;\ \Xi \vDash P \geqslant Q \dashv C_1$ and $\Theta;\ \Xi \vDash N' \leqslant M' \dashv C_2$ and returns $\Xi \vdash C_1 \,\&\, C_2 = C$ as the result.

By the soundness of constraint merge (Lemma 89), $\Xi \vdash \widehat{\sigma} : C_1$ and $\Xi \vdash \widehat{\sigma} : C_2$. Then by the soundness of positive subtyping (Lemma 86), $\Theta \vdash [\widehat{\sigma}]P \geqslant Q$; and by the induction hypothesis, $\Theta \vdash [\widehat{\sigma}]N' \leqslant M'$. This way, by $(\to^{\leqslant})$, $\Theta \vdash [\widehat{\sigma}](P \to N') \leqslant Q \to M'$.

Case 2. $(\textsc{Var}^{\leqslant}_{-})$, and then $\Theta;\ \Xi \vDash N \leqslant M \dashv C$ has shape $\Theta;\ \Xi \vDash \alpha^{-} \leqslant \alpha^{-} \dashv \cdot$
This case is symmetric to case 2 of Lemma 86.

Case 3. $(\uparrow^{\leqslant})$, and then $\Theta;\ \Xi \vDash N \leqslant M \dashv C$ has shape $\Theta;\ \Xi \vDash \uparrow P \leqslant \uparrow Q \dashv C$
This case is symmetric to case 3 of Lemma 86.

Case 4. $(\forall^{\leqslant})$, and then $\Theta;\ \Xi \vDash N \leqslant M \dashv C$ has shape $\Theta;\ \Xi \vDash \forall\overrightarrow{\alpha^{+}}.\ N' \leqslant \forall\overrightarrow{\beta^{+}}.\ M' \dashv C$ s.t. either $\overrightarrow{\alpha^{+}}$ or $\overrightarrow{\beta^{+}}$ is not empty
This case is symmetric to case 4 of Lemma 86.

■

Lemma 93 (Completeness of the Negative Subtyping). *Suppose that* $\Theta \vdash^{\supseteq} \Xi$, $\Theta \vdash M$, $\Theta\,;\mathtt{dom}\,(\Xi) \vdash N$, *and* N *does not contain negative unification variables* ($\widehat{\alpha}^{-} \notin \mathtt{fav}N$). *Then for any* $\Xi \vdash \widehat{\sigma} : \mathtt{fav}(N)$ *such that* $\Theta \vdash [\widehat{\sigma}]N \leqslant M$, *there exists* $\Theta;\ \Xi \vDash N \leqslant M \dashv C$ *and moreover,* $\Xi \vdash \widehat{\sigma} : C$.

Proof We prove it by induction on $\Theta \vdash [\widehat{\sigma}]N \leqslant M$. Let us consider the last rule used in the derivation of $\Theta \vdash [\widehat{\sigma}]N \leqslant M$.

Case 1. $\Theta \vdash [\widehat{\sigma}]N \leqslant M$ is derived by $(\uparrow^{\leqslant})$
Then $N = \uparrow P$, since the substitution $[\widehat{\sigma}]N$ must preserve the top-level constructor of $N \neq \widehat{\alpha}^{-}$ (since by assumption, $\widehat{\alpha}^{-} \notin \mathtt{fav}N$), and $Q = \downarrow M$, and by inversion, $\Theta \vdash [\widehat{\sigma}]N \simeq^{\leqslant} M$. The rest of the proof is symmetric to case 3 of Lemma 87: notice that

the algorithm does not make a recursive call, and the difference in the induction statement for the positive and the negative case here does not matter.

Case 2. $\Theta \vdash [\widehat{\sigma}]N \leqslant M$ is derived by $(\to^{\leqslant})$, i.e. $[\widehat{\sigma}]N = [\widehat{\sigma}]P \to [\widehat{\sigma}]N'$ and $M = Q \to M'$, and by inversion, $\Theta \vdash [\widehat{\sigma}]P \geqslant Q$ and $\Theta \vdash [\widehat{\sigma}]N' \leqslant M'$.

The algorithm makes two recursive calls: $\Theta; \Xi \vDash P \geqslant Q \dashv C_1$ and $\Theta; \Xi \vDash N' \leqslant M' \dashv C_2$, and then returns $\Xi \vdash C_1 \,\&\, C_2 = C$ as the result. Let us show that these recursive calls are successful and the returning constraints are fulfilled by $\widehat{\sigma}$.

Notice that from the inversion of $\Theta \vdash M$, we have: $\Theta \vdash Q$ and $\Theta \vdash M'$; from the inversion of $\Theta\,;\mathtt{dom}\,(\Xi) \vdash N$, we have: $\Theta\,;\mathtt{dom}\,(\Xi) \vdash P$ and $\Theta\,;\mathtt{dom}\,(\Xi) \vdash N'$; and since N does not contain negative unification variables, N' does not contain negative unification variables either.

This way, we can apply the induction hypothesis to $\Theta \vdash [\widehat{\sigma}]N' \leqslant M'$ to obtain $\Theta; \Xi \vDash N' \leqslant M' \dashv C_2$ such that $\Xi \vdash C_2 : \mathtt{fav}(N')$ and $\Xi \vdash \widehat{\sigma} : C_2$. Also, we can apply the completeness of the positive subtyping (Lemma 87) to $\Theta \vdash [\widehat{\sigma}]P \geqslant Q$ to obtain $\Theta; \Xi \vDash P \geqslant Q \dashv C_1$ such that $\Xi \vdash C_1 : \mathtt{fav}(P)$ and $\Xi \vdash \widehat{\sigma} : C_1$.

Finally, we need to show that the merge of C_1 and C_2 is successful and satisfies the required properties. To do so, we apply the completeness of subtyping constraint merge (Lemma 91) (notice that $\Xi \vdash \widehat{\sigma} : \mathtt{fav}(P \to N')$ means $\Xi \vdash \widehat{\sigma} : \mathtt{fav}(P) \cup \mathtt{fav}(N')$). This way, $\Xi \vdash C_1 \,\&\, C_2 = C$ is defined and $\Xi \vdash \widehat{\sigma} : C$ holds.

Case 3. $\Theta \vdash [\widehat{\sigma}]N \leqslant M$ is derived by $(\forall^{\leqslant})$. Since N does not contain negative unification variables, N must be of the form $\forall\overrightarrow{\alpha^{+}}.\, N'$, such that $[\widehat{\sigma}]N = \forall\overrightarrow{\alpha^{+}}.\, [\widehat{\sigma}]N'$ and $[\widehat{\sigma}]N' \neq \forall \ldots$ (assuming $\overrightarrow{\alpha^{+}}$ does not intersect with the range of $\widehat{\sigma}$). Also, $M = \forall\overrightarrow{\beta^{+}}.\, M'$ and either $\overrightarrow{\alpha^{+}}$ or $\overrightarrow{\beta^{+}}$ is non-empty.

The rest of the proof is symmetric to case 4 of Lemma 87. To apply the induction hypothesis, we need to show additionally that there are no negative unification variables in $N_0 = [\overrightarrow{\widehat{\alpha}^{+}}/\overrightarrow{\alpha^{+}}]N'$. This is because $\mathtt{fav} N_0 \subseteq \mathtt{fav} N \cup \overrightarrow{\widehat{\alpha}^{+}}$, and N is free of negative unification variables by assumption.

Case 4. $\Theta \vdash [\widehat{\sigma}]N \leqslant M$ is derived by $(\mathrm{Var}^{\leqslant}_{-})$.
Then $N = [\widehat{\sigma}]N = \alpha^{-} = M$. Here the first equality holds because N is not a unification variable: by assumption, N is free of negative unification variables. The second and the third equations hold because $(\mathrm{Var}^{\leqslant}_{-})$ was applied.

The rest of the proof is symmetric to case 2 of Lemma 87.

■

C.4 Declarative Typing

Definition 30 (Number of prenex quantifiers)**.** *Let us define* $\mathsf{npq}(N)$ *and* $\mathsf{npq}(P)$ *as the number of prenex quantifiers in these types, i.e.*

+ $\mathsf{npq}(\exists\overrightarrow{\alpha^{-}}.\, P) = |\overrightarrow{\alpha^{-}}|$, *if* $P \neq \exists\overrightarrow{\beta^{-}}.\, P'$,
− $\mathsf{npq}(\forall\overrightarrow{\alpha^{+}}.\, N) = |\overrightarrow{\alpha^{+}}|$, *if* $N \neq \forall\overrightarrow{\beta^{+}}.\, N'$.

Definition 31 (Size of a Declarative Judgement). *For a declarative typing judgment J let us define a metrics* $\mathsf{size}(J)$ *as a pair of numbers in the following way:*

- $\mathsf{size}(\Theta; \Gamma \vdash v\colon P) = (\mathsf{size}(v), 0)$;
- $\mathsf{size}(\Theta; \Gamma \vdash c\colon N) = (\mathsf{size}(c), 0)$;
- $\mathsf{size}(\Theta\,; \Gamma \vdash N \bullet \vec{v} \Rrightarrow M) = (\mathsf{size}(\vec{v}), \mathsf{npq}(N)))$

where $\mathsf{size}(v)$ *or* $\mathsf{size}(c)$ *is the size of the syntax tree of the term v or c and* $\mathsf{size}(\vec{v})$ *is the sum of sizes of the terms in* $\vec{v}$.

Definition 32 (Number of Equivalence Nodes). *For a tree T inferring a declarative typing judgment, let us define a function* $\mathsf{eq\_nodes}(T)$ *as the number of nodes in T labeled with* $(\simeq_+^{INF})$ *or* $(\simeq_-^{INF})$.

Definition 33 (Metric). *For a tree T inferring a declarative typing judgment J, let us define a metric* $\mathsf{metric}(T)$ *as a pair* $(\mathsf{size}(J), \mathsf{eq\_nodes}(T))$.

Lemma 57. *If* $\Theta\,; \Gamma \vdash N_1 \bullet \vec{v} \Rrightarrow M$ *and* $\Theta \vdash N_1 \simeq^{\leqslant} N_2$ *then* $\Theta\,; \Gamma \vdash N_2 \bullet \vec{v} \Rrightarrow M$.

Proof By Lemma 34, $\Theta \vdash N_1 \simeq^{\leqslant} N_2$ implies $N_1 \simeq^{D} N_2$. Let us prove the required judgement by induction on $N_1 \simeq^{D} N_2$. Let us consider the last rule used in the derivation.

Case 1. $(\textsc{Var}_-^{\simeq^D})$. It means that N_1 is α^- and N_2 is α^-. Then the required property coincides with the assumption.

Case 2. $(\uparrow^{\simeq^D})$. It means that N_1 is $\uparrow P_1$ and N_2 is $\uparrow P_2$. where $P_1 \simeq^{D} P_2$.

Then the only rule applicable to infer $\Theta\,; \Gamma \vdash \uparrow P_1 \bullet \vec{v} \Rrightarrow M$ is $(\emptyset^{\mathrm{INF}}_{\bullet\Rrightarrow})$, meaning that $\vec{v} =$ and $\Theta \vdash \uparrow P_1 \simeq^{\leqslant} M$. Then by transitivity of equivalence Corollary 10, $\Theta \vdash \uparrow P_2 \simeq^{\leqslant} M$, and then $(\emptyset^{\mathrm{INF}}_{\bullet\Rrightarrow})$ is applicable to infer $\Theta\,; \Gamma \vdash \uparrow P_2 \bullet \Rrightarrow M$.

Case 3. $(\to^{\simeq^D})$. Then we are proving that $\Theta\,; \Gamma \vdash (Q_1 \to N_1) \bullet v, \vec{v} \Rrightarrow M$ and $Q_1 \to N_1 \simeq^{D} Q_2 \to N_2$ imply $\Theta\,; \Gamma \vdash (Q_2 \to N_2) \bullet v, \vec{v} \Rrightarrow M$.

By inversion, $(Q_1 \to N_1) \simeq^{D} (Q_2 \to N_2)$ means $Q_1 \simeq^{D} Q_2$ and $N_1 \simeq^{D} N_2$.

By inversion of $\Theta\,; \Gamma \vdash (Q_1 \to N_1) \bullet v, \vec{v} \Rrightarrow M$:

1. $\Theta; \Gamma \vdash v\colon P$
2. $\Theta \vdash Q_1 \geqslant P$, and then by transitivity Lemma 24, $\Theta \vdash Q_2 \geqslant P$;
3. $\Theta\,; \Gamma \vdash N_1 \bullet \vec{v} \Rrightarrow M$, and then by induction hypothesis, $\Theta\,; \Gamma \vdash N_2 \bullet \vec{v} \Rrightarrow M$.

Since we have $\Theta; \Gamma \vdash v\colon P$, $\Theta \vdash Q_2 \geqslant P$ and $\Theta\,; \Gamma \vdash N_2 \bullet \vec{v} \Rrightarrow M$, we can apply $(\to^{\mathrm{INF}}_{\bullet\Rrightarrow})$ to infer $\Theta\,; \Gamma \vdash (Q_2 \to N_2) \bullet v, \vec{v} \Rrightarrow M$.

Case 4. $(\forall^{\simeq^D})$ Then we are proving that $\Theta\,; \Gamma \vdash \forall \overrightarrow{\alpha^+}_1.\, N_1' \bullet \vec{v} \Rrightarrow M$ and $\forall \overrightarrow{\alpha^+}_1.\, N_1' \simeq^{D} \forall \overrightarrow{\alpha^+}_2.\, N_2'$ imply $\Theta\,; \Gamma \vdash \forall \overrightarrow{\alpha^+}_2.\, N_2' \bullet \vec{v} \Rrightarrow M$.

By inversion of $\forall \overrightarrow{\alpha^+}_1.\, N_1' \simeq^{D} \forall \overrightarrow{\alpha^+}_2.\, N_2'$:

1. $\overrightarrow{\alpha^+}_2 \cap \mathtt{fv}\, N_1 = \emptyset$,
2. there exists a bijection $\mu : (\overrightarrow{\alpha^+}_2 \cap \mathtt{fv}\, N_2') \leftrightarrow (\overrightarrow{\alpha^+}_1 \cap \mathtt{fv}\, N_1')$ such that $N_1' \simeq^{D} [\mu] N_2'$.

By inversion of $\Theta ; \Gamma \vdash \forall \overrightarrow{\alpha^+}_1 . N'_1 \bullet \vec{v} \Longrightarrow M$:

1. $\Theta \vdash \sigma : \overrightarrow{\alpha^+}_1$
2. $\Theta ; \Gamma \vdash [\sigma] N'_1 \bullet \vec{v} \Longrightarrow M$
3. $\vec{v} \neq$

Let us construct $\Theta \vdash \sigma_0 : \overrightarrow{\alpha^+}_2$ in the following way:

$$\begin{cases} [\sigma_0]\alpha^+ = [\sigma][\mu]\alpha^+ & \text{if } \alpha^+ \in \overrightarrow{\alpha^+}_2 \cap \mathtt{fv}\, N'_2 \\ [\sigma_0]\alpha^+ = \exists \beta^- . \downarrow \beta^- & \text{otherwise (the type does not matter here)} \end{cases}$$

Then to infer $\Theta ; \Gamma \vdash N_2 \bullet \vec{v} \Longrightarrow M$, we apply $(\to^{\mathrm{INF}}_{\bullet\Longrightarrow})$ with σ_0. Let us show the required premises:

1. $\Theta \vdash \sigma_0 : \overrightarrow{\alpha^+}_2$ by construction;
2. $\vec{v} \neq$ as noted above;
3. To show $\Theta ; \Gamma \vdash [\sigma_0] N'_2 \bullet \vec{v} \Longrightarrow M$, Notice that $[\sigma_0]N'_2 = [\sigma][\mu]N'_2$ and since $[\mu]N'_2 \simeq^D N'_1$, $[\sigma][\mu]N'_2 \simeq^D [\sigma]N'_1$. This way, by Lemma 29, $\Theta \vdash [\sigma]N'_1 \simeq^{\leqslant} [\sigma_0]N'_2$. Then the required judgement holds by the induction hypothesis applied to $\Theta ; \Gamma \vdash [\sigma] N'_1 \bullet \vec{v} \Longrightarrow M$.

■

Lemma 50 (Declarative typing is preserved under context equivalence)**.** *Assuming* $\Theta \vdash \Gamma_1$, $\Theta \vdash \Gamma_2$, *and* $\Theta \vdash \Gamma_1 \simeq^{\leqslant} \Gamma_2$*:*

+ *for any tree* T_1 *inferring* $\Theta ; \Gamma_1 \vdash v : P$, *there exists a tree* T_2 *inferring* $\Theta ; \Gamma_2 \vdash v : P$.
− *for any tree* T_1 *inferring* $\Theta ; \Gamma_1 \vdash c : N$, *there exists a tree* T_2 *inferring* $\Theta ; \Gamma_2 \vdash c : N$.
• *for any tree* T_1 *inferring* $\Theta ; \Gamma_1 \vdash N \bullet \vec{v} \Longrightarrow M$, *there exists a tree* T_2 *inferring* $\Theta ; \Gamma_2 \vdash N \bullet \vec{v} \Longrightarrow M$.

Proof Let us prove it by induction on the $\mathsf{metric}(T_1)$. Let us consider the last rule applied in T_1 (i.e., its root node).

Case 1. $(\textsc{Var}^{\mathrm{INF}})$
Then we are proving that $\Theta ; \Gamma_1 \vdash x : P$ implies $\Theta ; \Gamma_2 \vdash x : P$. By inversion, $x : P \in \Gamma_1$, and since $\Theta \vdash \Gamma_1 \simeq^{\leqslant} \Gamma_2$, $x : P' \in \Gamma_2$ for some P' such that $\Theta \vdash P \simeq^{\leqslant} P'$. Then we infer $\Theta ; \Gamma_2 \vdash x : P'$ by $(\textsc{Var}^{\mathrm{INF}})$, and next, $\Theta ; \Gamma_2 \vdash x : P$ by $(\simeq^{\mathrm{INF}}_{+})$.

Case 2. For $(\{\}^{\mathrm{INF}})$, $(\textsc{ann}^{\mathrm{INF}}_{+})$, (Λ^{INF}), $(\textsc{ret}^{\mathrm{INF}})$, and $(\textsc{ann}^{\mathrm{INF}}_{-})$ the proof is analogous. We apply the induction hypothesis to the premise of the rule to substitute Γ_1 for Γ_2 in it. The induction is applicable because the metric of the premises is less than the metric of the conclusion: the term in the premise is a syntactic subterm of the term in the conclusion.

And after that, we apply the same rule to infer the required judgment.

Case 3. $(\simeq^{\text{INF}}_{+})$ and $(\simeq^{\text{INF}}_{-})$ In these cases, the induction hypothesis is also applicable to the premise: although the first component of the metric is the same for the premise and the conclusion: $\mathsf{size}(\Theta;\Gamma\vdash c\colon N') = \mathsf{size}(\Theta;\Gamma\vdash c\colon N) = \mathsf{size}(c)$, the second component of the metric is less for the premise by one, since the equivalence rule was applied to turn the premise tree into $T1$. Having made this note, we continue the proof in the same way as in the previous case.

Case 4. (λ^{INF}) Then we prove that $\Theta;\Gamma_1\vdash \lambda x\colon P.\, c\colon P\to N$ implies $\Theta;\Gamma_2\vdash \lambda x\colon P.\, c\colon P\to N$. Analogously to the previous cases, we apply the induction hypothesis to the equivalent contexts $\Theta\vdash\Gamma_1, x\colon P \simeq^{\leqslant} \Gamma_2, x\colon P$ and the premise $\Theta;\Gamma_1, x\colon P\vdash c\colon N$ to obtain $\Theta;\Gamma_2, x\colon P\vdash c\colon N$. Notice that c is a subterm of $\lambda x\colon P.\, c$, i.e., the metric of the premise tree is less than the metric of the conclusion, and the induction hypothesis is applicable. Then we infer $\Theta;\Gamma_2\vdash \lambda x\colon P.\, c\colon P\to N$ by (λ^{INF}).

Case 5. $(\text{LET}^{\text{INF}})$ Then we prove that $\Theta;\Gamma_1\vdash \mathtt{let}\, x = v;\ c\colon N$ implies $\Theta;\Gamma_2\vdash \mathtt{let}\, x = v;\ c\colon N$. First, we apply the induction hypothesis to $\Theta;\Gamma_1\vdash v\colon P$ to obtain $\Theta;\Gamma_2\vdash v\colon P$ of the same size.

Then we apply the induction hypothesis to the equivalent contexts $\Theta\vdash\Gamma_1, x\colon P \simeq^{\leqslant} \Gamma_2, x\colon P$ and the premise $\Theta;\Gamma_1, x\colon P\vdash c\colon N$ to obtain $\Theta;\Gamma_2, x\colon P\vdash c\colon N$. Then we infer $\Theta;\Gamma_2\vdash \mathtt{let}\, x = v;\ c\colon N$ by $(\text{LET}^{\text{INF}})$.

Case 6. $(\text{LET}^{\text{INF}}_{\text{C}})$ Then we prove that $\Theta;\Gamma_1\vdash \mathtt{let}\, x\colon P = c;\ c'\colon N$ implies $\Theta;\Gamma_2\vdash \mathtt{let}\, x\colon P = c;\ c'\colon N$.

First, we apply the induction hypothesis to $\Theta;\Gamma_1\vdash c\colon M$ to obtain $\Theta;\Gamma_2\vdash c\colon M$ of the same size.

Then we apply the induction hypothesis to the equivalent contexts $\Theta\vdash\Gamma_1, x\colon P \simeq^{\leqslant} \Gamma_2, x\colon P$ and the premise $\Theta;\Gamma_1, x\colon P\vdash c'\colon N$ to obtain $\Theta;\Gamma_2, x\colon P\vdash c'\colon N$. Then we infer $\Theta;\Gamma_2\vdash \mathtt{let}\, x\colon P = c;\ c'\colon N$ by $(\text{LET}^{\text{INF}}_{\text{C}})$.

Case 7. $(\text{LET}^{\text{INF}}_{@})$ Then we prove that $\Theta;\Gamma_1\vdash \mathtt{let}\, x = v(\vec{v});\ c\colon N$ implies $\Theta;\Gamma_2\vdash \mathtt{let}\, x = v(\vec{v});\ c\colon N$.

We apply the induction hypothesis to each of the premises. to rewrite:

- $\Theta;\Gamma_1\vdash v\colon \downarrow M$ into $\Theta;\Gamma_2\vdash v\colon \downarrow M$,
- $\Theta\,;\Gamma_1\vdash M\bullet\vec{v}\Rrightarrow\uparrow Q$ into $\Theta\,;\Gamma_2\vdash M\bullet\vec{v}\Rrightarrow\uparrow Q$.
- $\Theta;\Gamma_1, x\colon Q\vdash c\colon N$ into $\Theta;\Gamma_2, x\colon Q\vdash c\colon N$ (notice that $\Theta\vdash\Gamma_1, x\colon Q \simeq^{\leqslant} \Gamma_2, x\colon Q$).

It is left to show the principality of $\Theta\,;\Gamma_2\vdash M\bullet\vec{v}\Rrightarrow\uparrow Q$. Let us assume that this judgment holds for other Q', i.e. there exists a tree T_0 inferring $\Theta\,;\Gamma_2\vdash M\bullet\vec{v}\Rrightarrow > \uparrow Q'$. Then notice that the induction hypothesis applies to T_0: the first component of the first component of $\mathsf{metric}(T_0)$ is $S = \sum_{v\in\vec{v}}\mathsf{size}(v)$, and it is less than the corresponding component of $\mathsf{metric}(T_1)$, which is $\mathsf{size}(\mathtt{let}\, x = v(\vec{v});\ c) = 1 + \mathsf{size}(v) + \mathsf{size}(c) + S$. This way, $\Theta\,;\Gamma_1\vdash M\bullet\vec{v}\Rrightarrow\uparrow Q'$ holds by the induction hypothesis, but since $\Theta\,;\Gamma_1\vdash M\bullet\vec{v}\Rrightarrow\uparrow Q$ is principal, we have $\Theta\vdash Q'\geqslant Q$. This

way, $\Theta\,;\Gamma_2 \vdash M \bullet \vec{v} \Rightarrow\!\!> \uparrow Q$ principal. Then we infer $\Theta; \Gamma_2 \vdash \mathtt{let}\, x = v(\vec{v});\ c\colon N$ by $(\text{LET}^{\text{INF}}_{@})$.

Case 8. $(\text{LET}^{\text{INF}}_{:@})$ Then we prove that $\Theta; \Gamma_1 \vdash \mathtt{let}\, x : P = v(\vec{v});\ c\colon N$ implies $\Theta; \Gamma_2 \vdash \mathtt{let}\, x : P = v(\vec{v});\ c\colon N$.

As in the previous case, we apply the induction hypothesis to each of the premises and rewrite:

- $\Theta; \Gamma_1 \vdash v\colon \downarrow M$ into $\Theta; \Gamma_2 \vdash v\colon \downarrow M$,
- $\Theta\,;\Gamma_1 \vdash M \bullet \vec{v} \Rightarrow\!\!> M'$ into $\Theta\,;\Gamma_2 \vdash M \bullet \vec{v} \Rightarrow\!\!> M'$, and
- $\Theta; \Gamma_1, x : P \vdash c\colon N$ into $\Theta; \Gamma_2, x : P \vdash c\colon N$ (notice that $\Theta \vdash \Gamma_1, x : P \simeq^{\leqslant} \Gamma_2, x : P$).

Notice that $\Theta \vdash P$ and $\Theta \vdash M' \leqslant \uparrow P$ do not depend on the variable context, and hold by assumption. Then we infer $\Theta; \Gamma_2 \vdash \mathtt{let}\, x : P = v(\vec{v});\ c\colon N$ by $(\text{LET}^{\text{INF}}_{:@})$.

Case 9. $(\text{LET}^{\text{INF}}_{\exists})$, and $(\text{ANN}^{\text{INF}}_{-})$ are proved in the same way.

Case 10. $(\emptyset^{\text{INF}}_{\bullet\Rightarrow\!\!>})$ Then we are proving that $\Theta\,;\Gamma_1 \vdash N\bullet \Rightarrow\!\!> N'$ (inferred by $(\emptyset^{\text{INF}}_{\bullet\Rightarrow\!\!>})$) implies $\Theta\,;\Gamma_2 \vdash N\bullet \Rightarrow\!\!> N'$.

To infer $\Theta\,;\Gamma_2 \vdash N\bullet \Rightarrow\!\!> N'$, we apply $(\emptyset^{\text{INF}}_{\bullet\Rightarrow\!\!>})$, noting that $\Theta \vdash N \simeq^{\leqslant} N'$ holds by assumption.

Case 11. $(\to^{\text{INF}}_{\bullet\Rightarrow\!\!>})$ Then we are proving that $\Theta\,;\Gamma_1 \vdash Q \to N \bullet v, \vec{v} \Rightarrow\!\!> M$ (inferred by $(\to^{\text{INF}}_{\bullet\Rightarrow\!\!>})$) implies $\Theta\,;\Gamma_2 \vdash Q \to N \bullet v, \vec{v} \Rightarrow\!\!> M$. And uniqueness of the M in the first case implies uniqueness in the second case.

By induction, we rewrite $\Theta; \Gamma_1 \vdash v\colon P$ into $\Theta; \Gamma_2 \vdash v\colon P$, and $\Theta\,;\Gamma_1 \vdash N \bullet \vec{v} \Rightarrow\!\!> M$ into $\Theta\,;\Gamma_2 \vdash N \bullet \vec{v} \Rightarrow\!\!> M$. Then we infer $\Theta\,;\Gamma_2 \vdash Q \to N \bullet v, \vec{v} \Rightarrow\!\!> M$ by $(\to^{\text{INF}}_{\bullet\Rightarrow\!\!>})$.

Now, let us show the uniqueness. The only rule that can infer $\Theta\,;\Gamma_1 \vdash Q \to N \bullet v, \vec{v} \Rightarrow\!\!> M$ is $(\to^{\text{INF}}_{\bullet\Rightarrow\!\!>})$. Then by inversion, uniqueness of $\Theta\,;\Gamma_1 \vdash Q \to N \bullet v, \vec{v} \Rightarrow\!\!> M$ implies uniqueness of $\Theta\,;\Gamma_1 \vdash N \bullet \vec{v} \Rightarrow\!\!> M$. By the induction hypothesis, it implies the uniqueness of $\Theta\,;\Gamma_2 \vdash N \bullet \vec{v} \Rightarrow\!\!> M$.

Suppose that $\Theta\,;\Gamma_2 \vdash Q \to N \bullet v, \vec{v} \Rightarrow\!\!> M'$. By inversion, $\Theta\,;\Gamma_2 \vdash N \bullet \vec{v} \Rightarrow\!\!> M'$, which by uniqueness of $\Theta\,;\Gamma_2 \vdash N \bullet \vec{v} \Rightarrow\!\!> M$ implies $\Theta \vdash M \simeq^{\leqslant} M'$.

Case 12. $(\forall^{\text{INF}}_{\bullet\Rightarrow\!\!>})$ Then we are proving that $\Theta\,;\Gamma_1 \vdash \forall\overrightarrow{\alpha^{+}}.\, N \bullet \vec{v} \Rightarrow\!\!> M$ (inferred by $(\forall^{\text{INF}}_{\bullet\Rightarrow\!\!>})$) implies $\Theta\,;\Gamma_2 \vdash \forall\overrightarrow{\alpha^{+}}.\, N \bullet \vec{v} \Rightarrow\!\!> M$.

By inversion, we have σ such that $\Theta \vdash \sigma : \overrightarrow{\alpha^{+}}$ and $\Theta\,;\Gamma_1 \vdash [\sigma]N \bullet \vec{v} \Rightarrow\!\!> M$ is inferred. Let us denote the inference tree as T_1'. Notice that the induction hypothesis is applicable to T_1': $\mathsf{metric}(T_1') = ((\mathsf{size}(\vec{v}), 0), x)$ is less than $\mathsf{metric}(T_1) = ((\mathsf{size}(\vec{v}), |\overrightarrow{\alpha^{+}}|), y)$ for any x and y, since $|\overrightarrow{\alpha^{+}}| > 0$ by inversion.

This way, by the induction hypothesis, there exists a tree T_2' inferring $\Theta\,;\Gamma_2 \vdash [\sigma]N \bullet \vec{v} \Rightarrow\!\!> M$. Notice that the premises $\vec{v} \neq \cdot$, $\Theta \vdash \sigma : \overrightarrow{\alpha^{+}}$, and $\overrightarrow{\alpha^{+}} \neq \cdot$ do not depend on the variable context, and hold by inversion. Then we infer $\Theta\,;\Gamma_2 \vdash \forall\overrightarrow{\alpha^{+}}.\, N \bullet \vec{v} \Rightarrow\!\!> M$ by $(\forall^{\text{INF}}_{\bullet\Rightarrow\!\!>})$.

■

C.5 Algorithmic Typing

C.5.1 Singularity and Minimal Instantiation

Lemma 94 (Soundness of Minimal Instantiation)**.** *Suppose that* $\Theta \vdash^{\supseteq} \Xi$, $\Xi \vdash C$, *and* Θ; $\mathtt{dom}(\Xi) \vdash P$. *If* P *is* C*-minimized by* $\widehat{\sigma}$ *then*

- $\Xi \vdash \widehat{\sigma} : \mathtt{fav} P$,
- $\Xi \vdash \widehat{\sigma} : C$,
- $\widehat{\sigma}$ *is normalized, and*
- *for any other* $\Xi \vdash \widehat{\sigma}' : \mathtt{fav} P$ *respecting* C *(i.e.,* $\Xi \vdash \widehat{\sigma}' : C$*), we have* $\Theta \vdash [\widehat{\sigma}']P \geqslant [\widehat{\sigma}]P$.

Proof We prove it by induction on the inference of P is C-minimized by $\widehat{\sigma}$. Let us consider the last rule used in the inference.

Case 1. (UVAR^{MIN}), which means that the inferred judgment is $\widehat{\alpha}^+$ is C-minimized by $(\mathtt{nf}\,(P)/\widehat{\alpha}^+)$, and by inversion, $(\widehat{\alpha}^+ :\geqslant P) \in C$. Let us show the required properties:

- $\Xi \vdash (\mathtt{nf}\,(P)/\widehat{\alpha}^+) : \mathtt{fav}\widehat{\alpha}^+$ holds trivially;
- $\Xi \vdash (\mathtt{nf}\,(P)/\widehat{\alpha}^+) : C$ holds since $\Theta \vdash \mathtt{nf}\,(P) : (\widehat{\alpha}^+ :\geqslant P)$, which is true since $\Theta \vdash \mathtt{nf}\,(P) \geqslant P$ by the soundness of normalization (Lemma 41);
- $(\mathtt{nf}\,(P)/\widehat{\alpha}^+)$ is normalized trivially;
- let us take an arbitrary $\Xi \vdash \widehat{\sigma}' : \widehat{\alpha}^+$ respecting C. Since $\widehat{\sigma}'$ respects C, $\Theta \vdash [\widehat{\sigma}']\widehat{\alpha}^+ \geqslant P$ holds, and then $\Theta \vdash [\widehat{\sigma}']\widehat{\alpha}^+ \geqslant \mathtt{nf}\,(P)$ holds by the soundness of normalization and transitivity of subtyping. Finally, $\Theta \vdash [\widehat{\sigma}']\widehat{\alpha}^+ \geqslant \mathtt{nf}\,(P)$ can be rewritten as $\Theta \vdash [\widehat{\sigma}']\widehat{\alpha}^+ \geqslant [(\mathtt{nf}\,(P)/\widehat{\alpha}^+)]\widehat{\alpha}^+$.

Case 2. ($\exists^{\text{MIN}}$), which means that the inferred judgment has form $\exists\overrightarrow{\alpha}. P$ is C-minimized by $\widehat{\sigma}$, and by inversion, P is C-minimized by $\widehat{\sigma}$. By applying the induction hypothesis to P is C-minimized by $\widehat{\sigma}$ we have

- $\Xi \vdash \widehat{\sigma} : \mathtt{fav} P$, which also means $\Xi \vdash \widehat{\sigma} : \mathtt{fav}\exists\overrightarrow{\alpha}. P$,
- $\Xi \vdash \widehat{\sigma} : C$,
- $\widehat{\sigma}$ is normalized, and
- for any other $\Xi \vdash \widehat{\sigma}' : \mathtt{fav} P$ respecting C (i.e., $\Xi \vdash \widehat{\sigma}' : C$), we have $\Theta, \overrightarrow{\alpha} \vdash [\widehat{\sigma}']P \geqslant [\widehat{\sigma}]P$, which immediately implies $\Theta \vdash [\widehat{\sigma}']\exists\overrightarrow{\alpha}. P \geqslant [\widehat{\sigma}]\exists\overrightarrow{\alpha}. P$ (the left-hand side existential variables are instantiated with the corresponding right-hand side existential variables).

Case 3. (SING^{MIN}), which means that the inferred judgment has form P is C-minimized by $\widehat{\sigma}$, and by inversion, $\mathtt{fav}(P) \subseteq \mathtt{dom}\,(C)$ and

$C|_{\mathtt{fav}(P)}$ $\mathtt{singular\ with}\ \widehat{\sigma}$. Let us apply the soundness of singularity (Lemma 98) to $C|_{\mathtt{fav}(P)}$ $\mathtt{singular\ with}\ \widehat{\sigma}$ to obtain the following properties:

- $\Xi \vdash \widehat{\sigma} : \mathtt{fav}(P) \cap \mathtt{dom}\,(C)$, which also means $\Xi \vdash \widehat{\sigma} : \mathtt{fav}(P)$,
- $\Xi \vdash \widehat{\sigma} : C|_{\mathtt{fav}(P)}$,
- $\widehat{\sigma}$ is normalized, and
- for any other $\Xi \vdash \widehat{\sigma}' : \mathtt{fav}(P)$ respecting $C|_{\mathtt{fav}(P)}$, we have $\Xi \vdash \widehat{\sigma}' \simeq^{\leqslant} \widehat{\sigma} : \mathtt{fav}(P)$. The latter means that $\Theta \vdash [\widehat{\sigma}']P \simeq^{\leqslant} [\widehat{\sigma}]P$, and in particular, $\Theta \vdash [\widehat{\sigma}']P \geqslant [\widehat{\sigma}]P$.

■

Lemma 95 (Completeness of Minimal Instantiation)**.** *Suppose that* $\Theta \vdash^{\supseteq} \Xi$, $\Xi \vdash C$, $\Theta\,;\mathtt{dom}\,(\Xi) \vdash P$, *and there exists* $\Xi \vdash \widehat{\sigma} : \mathtt{fav}P$ *respecting* C $(\Xi \vdash \widehat{\sigma} : C)$ *such that for any other* $\Xi \vdash \widehat{\sigma}' : \mathtt{fav}P$ *respecting* C $(\Xi \vdash \widehat{\sigma}' : C)$, *we have* $\Theta \vdash [\widehat{\sigma}']P \geqslant [\widehat{\sigma}]P$. *Then* P *is* C*-minimized by* $\mathtt{nf}\,(\widehat{\sigma})$.

Proof We prove it by induction on P.

Case 1. $P = \widehat{\alpha}^+$. Suppose that $\widehat{\alpha}^+ \notin \mathtt{dom}\,(C)$. Then the instantiation of $\widehat{\alpha}^+$ is not restricted, and thus, any type can instantiate it. However, among unrestricted instantiations, there is no minimum: any type P is *not* a subtype of $\downarrow\uparrow P$, which contradicts the assumption. This way, $\widehat{\alpha}^+ \in \mathtt{dom}\,(C)$.

If the entry restricting $\widehat{\alpha}^+$ in C is a *subtyping* entry $((\widehat{\alpha}^+ :\geqslant Q) \in C)$, then we apply (UVar$^{\mathrm{MIN}}$) to infer $\widehat{\alpha}^+$ is C-minimized by $(\mathtt{nf}\,(Q)/\widehat{\alpha}^+)$. It is left to show that $\mathtt{nf}\,(Q) = \mathtt{nf}\,([\widehat{\sigma}]\widehat{\alpha}^+)$. Since $\Xi \vdash \widehat{\sigma} : C$, and $(\widehat{\alpha}^+ :\geqslant Q) \in C$, we know that $\Theta \vdash [\widehat{\sigma}]\widehat{\alpha}^+ \geqslant Q$. On the other hand, let us consider $\Xi \vdash \widehat{\sigma}' : C$, that copies $\widehat{\sigma}$ on $\mathtt{dom}\,(C)$ except $\widehat{\alpha}^+$, where it is instantiated with Q. Then $\Theta \vdash [\widehat{\sigma}']\widehat{\alpha}^+ \geqslant [\widehat{\sigma}]\widehat{\alpha}^+$ means $\Theta \vdash Q \geqslant [\widehat{\sigma}]\widehat{\alpha}^+$, this way, $\Theta \vdash Q \simeq^{\leqslant} [\widehat{\sigma}]\widehat{\alpha}^+$, which by Lemma 48 means $\mathtt{nf}\,(Q) = \mathtt{nf}\,([\widehat{\sigma}]\widehat{\alpha}^+)$.

If the entry restricting $\widehat{\alpha}^+$ in C is an *equivalence* entry $((\widehat{\alpha}^+ :\simeq Q) \in C)$, then we wish to apply (sing$^{\mathrm{MIN}}$). The first premise $\mathtt{fav}(\widehat{\alpha}^+) \subseteq \mathtt{dom}\,(C)$ holds by assumption; to infer $C|_{\widehat{\alpha}^+}$ $\mathtt{singular\ with}\ \widehat{\sigma}_0$, we apply the completeness of singularity (Lemma 99). It applies because all the substitutions satisfying $C|_{\widehat{\alpha}^+} = (\widehat{\alpha}^+ :\simeq Q)$ are equivalent on $\widehat{\alpha}^+$ by transitivity of equivalence (Corollary 10): the satisfaction of this constraint means that the substitution sends $\widehat{\alpha}^+$ to Q or an equivalent type. This way, $C|_{\widehat{\alpha}^+}$ $\mathtt{singular\ with}\ \widehat{\sigma}_0$ for some $\widehat{\sigma}_0$, which means $\widehat{\alpha}^+$ is C-minimized by $\widehat{\sigma}_0$. To show that $\widehat{\sigma}_0 = \mathtt{nf}\,(\widehat{\sigma})$ notice that Since $\widehat{\sigma}_0$ is normalized and equivalent to $\widehat{\sigma}$ on $\widehat{\alpha}^+$, and only has $\widehat{\alpha}^+$ in its domain (by soundness of singularity, Lemma 98). This way, $\widehat{\alpha}^+$ is C-minimized by $\widehat{\sigma}$, as required.

Case 2. $P = \downarrow N$. Then since $\Theta \vdash \downarrow[\widehat{\sigma}']N \geqslant \downarrow[\widehat{\sigma}]N$ means $\Theta \vdash \downarrow[\widehat{\sigma}']N \simeq^{\leqslant} \downarrow[\widehat{\sigma}]N$ by inversion. Then by Lemma 10, $\widehat{\sigma}$ is equivalent to any other substitution $\Xi \vdash \widehat{\sigma}' : \mathtt{fav}N$ satisfying $C|_{\mathtt{fav}N}$, hence, the completeness of singularity (Lemma 99) can be applied to conclude that

- $\mathtt{fav}(N) = \mathtt{dom}\,(C|_{\mathtt{fav}N})$, then $\mathtt{fav}(P) \subseteq \mathtt{dom}\,(C)$,

- $C|_{\mathtt{fav}N}$ $\mathtt{singular\ with}\ \widehat{\sigma}_0$ for some (normalized) $\widehat{\sigma}_0$.

It means P is C-minimized by $\widehat{\sigma}_0$, and then since $\widehat{\sigma}_0$ is normalized and equivalent to $\widehat{\sigma}_0$ on $\mathtt{fav}(N)$, and its domain is $\mathtt{fav}(N)$, $\widehat{\sigma}_0 = \mathtt{nf}\,(\widehat{\sigma})$.

Case 3. $P = \exists\overrightarrow{\alpha^-}.\,\beta^+$ then as there are no algorithmic variables in P, $\mathtt{nf}\,([\widehat{\sigma}]P) = \beta^+$, and thus, we wish to show that $\exists\overrightarrow{\alpha^-}.\,\beta^+$ is C-minimized by $\cdot$. To do so, we apply $(\exists^{\mathrm{MIN}})$, and it is left to show that β^+ is C-minimized by $\cdot$, which holds vacuously by $(\mathrm{SING}^{\mathrm{MIN}})$.

Case 4. $P = \exists\overrightarrow{\alpha^-}.\,\widehat{\alpha}^+$ then $\Theta \vdash [\widehat{\sigma}']\exists\overrightarrow{\alpha^-}.\,\widehat{\alpha}^+ \geqslant [\widehat{\sigma}]\exists\overrightarrow{\alpha^-}.\,\widehat{\alpha}^+$ implies $\Theta \vdash \exists\overrightarrow{\alpha^-}.\,[\widehat{\sigma}']\widehat{\alpha}^+ \geqslant \exists\overrightarrow{\alpha^-}.\,[\widehat{\sigma}]\widehat{\alpha}^+$ implies $\Theta \vdash [\widehat{\sigma}']\widehat{\alpha}^+ \geqslant [\widehat{\sigma}]\widehat{\alpha}^+$. It means that $\widehat{\sigma}$ instantiates $\widehat{\alpha}^+$ to the minimal type among all the instantiations of $\widehat{\alpha}^+$ respecting C. In other words, we can apply the reasoning from case 1 to conclude that $\widehat{\alpha}^+$ is C-minimized by $\mathtt{nf}\,(\widehat{\sigma})$. And then $(\exists^{\mathrm{MIN}})$ gives us $\exists\overrightarrow{\alpha^-}.\,\widehat{\alpha}^+$ is C-minimized by $\mathtt{nf}\,(\widehat{\sigma})$.

Case 5. $P = \exists\overrightarrow{\alpha^-}.\,{\downarrow}N$ then $\Theta \vdash [\widehat{\sigma}']\exists\overrightarrow{\alpha^-}.\,{\downarrow}N \geqslant \exists\overrightarrow{\alpha^-}.\,{\downarrow}[\widehat{\sigma}]N$ implies $\Theta \vdash \exists\overrightarrow{\alpha^-}.\,{\downarrow}[\widehat{\sigma}']N \geqslant \exists\overrightarrow{\alpha^-}.\,{\downarrow}[\widehat{\sigma}]N$ implies $\Theta, \overrightarrow{\alpha^-} \vdash {\downarrow}[\sigma_0][\widehat{\sigma}']N \geqslant {\downarrow}[\widehat{\sigma}]N$ for some σ_0 implies $\Theta, \overrightarrow{\alpha^-} \vdash [\sigma_0][\widehat{\sigma}']N \simeq^{\leqslant} [\widehat{\sigma}]N$. By Lemma 10, it means in particular that $\widehat{\sigma}'$ and $\widehat{\sigma}$ are equivalent on $\mathtt{fav}N$. This way, we can apply the completeness of singularity (Lemma 99), and continue as in case 2 to conclude that ${\downarrow}N$ is C-minimized by $\mathtt{nf}\,(\widehat{\sigma})$. Then by $(\exists^{\mathrm{MIN}})$, we have $\exists\overrightarrow{\alpha^-}.\,{\downarrow}N$ is -minimized by $\mathtt{nf}\,(\widehat{\sigma})$.

■

Observation 17 (Minimal Instantiation is Deterministic)**.** *Suppose that* $\Theta \vdash^{\supseteq} \Xi$, $\Xi \vdash C$, $\Theta\,;\mathtt{dom}\,(\Xi) \vdash P$. *Then* P *is* C*-minimized by* $\widehat{\sigma}$ *and* P *is* C*-minimized by* $\widehat{\sigma}'$ *implies* $\widehat{\sigma} = \widehat{\sigma}'$.

Proof We prove it by induction on P is C-minimized by $\widehat{\sigma}$. It is easy to see that each inference rule is deterministic. ■

Lemma 96 (Soundness of Entry Singularity)**.**

+ *Suppose* $e\ \mathtt{singular\ with}\ P$ *for* P *well-formed in* Θ. *Then* $\Theta \vdash P : e$, P *is normalized, and for any* $\Theta \vdash P'$ *such that* $\Theta \vdash P' : e$, $\Theta \vdash P' \simeq^{\leqslant} P$;
- *Suppose* $e\ \mathtt{singular\ with}\ N$ *for* N *well-formed in* Θ. *Then* $\Theta \vdash N : e$, N *is normalized, and for any* $\Theta \vdash N'$ *such that* $\Theta \vdash N' : e$, $\Theta \vdash N' \simeq^{\leqslant} N$.

Proof Let us consider how $e\ \mathtt{singular\ with}\ P$ or $e\ \mathtt{singular\ with}\ N$ is formed.

Case 1. $(\simeq^{\mathrm{SING}}_{-})$, that is $e = \widehat{\alpha}^- :\simeq N_0$. and N is $\mathtt{nf}\,(N_0)$. Then $\Theta \vdash N' : e$ means $\Theta \vdash N' \simeq^{\leqslant} N_0$, (by inversion of $(:\simeq^{\mathrm{SAT}}_{-})$), which by transitivity, using Corollary 16, means $\Theta \vdash N' \simeq^{\leqslant} \mathtt{nf}\,(N_0)$, as required.

Case 2. $(\simeq^{\mathrm{SING}}_{+})$. This case is symmetric to the previous one.

Case 3. $(:\geqslant\alpha^{\mathrm{SING}})$, that is $e = \widehat{\alpha}^+ :\geqslant \exists\overrightarrow{\alpha^-}.\,\beta^+$, and $P = \beta^+$.

Since $\Theta \vdash \beta^+ \geqslant \exists\overrightarrow{\alpha^-}.\,\beta^+$, we have $\Theta \vdash \beta^+ : e$, as required.

Notice that $\Theta \vdash P' : e$ means $\Theta \vdash P' \geqslant \exists \overrightarrow{\alpha^-}.\, \beta^+$. Let us show that it implies $\Theta \vdash P' \simeq^{\leqslant} \beta^+$. By applying Lemma 77 once, we have $\Theta, \overrightarrow{\alpha^-} \vdash P' \geqslant \beta^+$. By applying it again, we notice that $\Theta, \overrightarrow{\alpha^-} \vdash P' \geqslant \beta^+$ implies $P_i = \exists \overrightarrow{\alpha^-}'.\, \beta^+$. Finally, it is easy to see that $\Theta \vdash \exists \overrightarrow{\alpha^-}'.\, \beta^+ \simeq^{\leqslant} \beta^+$

Case 4. $(:\geqslant\downarrow^{\text{SING}})$, that is $e = \widehat{\alpha^+} :\geqslant \exists \overrightarrow{\beta^-}.\downarrow N_1$, where $N_1 \simeq^D \beta^-{}_j$, and $P = \exists \alpha^-.\downarrow \alpha^-$.

Since $\Theta \vdash \exists \alpha^-.\downarrow \alpha^- \geqslant \exists \overrightarrow{\beta^-}.\downarrow N_1$ (by $(\exists^{\geqslant})$, with substitution N_1/α^-), we have $\Theta \vdash \exists \alpha^-.\downarrow \alpha^- : e$, as required.

Notice $\Theta \vdash P' : e$ means $\Theta \vdash P' \geqslant \exists \overrightarrow{\beta^-}.\downarrow N_1$. Let us show that it implies $\Theta \vdash P' \simeq^{\leqslant} \exists \alpha^-.\downarrow \alpha^-$.

$$
\begin{aligned}
\Theta \vdash P' \geqslant \exists \overrightarrow{\beta^-}.\downarrow N_1 &\Rightarrow \Theta \vdash \mathtt{nf}\,(P') \geqslant \exists \overrightarrow{\beta^-}'.\downarrow \mathtt{nf}\,(N_1) \\
&\quad (\text{where } \mathtt{ord}\,\overrightarrow{\beta^-}\ \mathtt{in}\ N' = \overrightarrow{\beta^-}') && \text{by Corollary 17} \\
&\Rightarrow \Theta \vdash \mathtt{nf}\,(P') \geqslant \exists \overrightarrow{\beta^-}'.\downarrow \mathtt{nf}\,(\beta^-{}_j) && \text{by Lemma 44} \\
&\Rightarrow \Theta \vdash \mathtt{nf}\,(P') \geqslant \exists \overrightarrow{\beta^-}'.\downarrow \beta^-{}_n && \text{by definition of normalization} \\
&\Rightarrow \Theta \vdash \mathtt{nf}\,(P') \geqslant \exists \beta^-{}_j.\downarrow \beta^-{}_j && \text{since } \mathtt{ord}\,\overrightarrow{\beta^-}\ \mathtt{in}\ \mathtt{nf}\,(N_1) = \beta^-{}_j \\
&\Rightarrow \Theta, \beta^-{}_j \vdash \mathtt{nf}\,(P') \geqslant \downarrow \beta^-{}_j \\
&\quad \text{and } \beta^-{}_j \notin \mathtt{fv}\,(\mathtt{nf}\,(P')) && \text{by Lemma 78}
\end{aligned}
$$

By Lemma 78, the last subtyping means that $\mathtt{nf}\,(P') = \exists \overrightarrow{\alpha^-}.\downarrow N'$, such that

1. $\Theta, \beta^-{}_j, \overrightarrow{\alpha^-} \vdash N'$
2. $\mathtt{ord}\,\overrightarrow{\alpha^-}\ \mathtt{in}\ N' = \overrightarrow{\alpha^-}$
3. for some substitution $\Theta, \beta^-{}_j \vdash \sigma : \overrightarrow{\alpha^-}$, $[\sigma]N' = \beta^-{}_j$.

Since $\beta^-{}_j \notin \mathtt{fv}\,(\mathtt{nf}\,(P'))$, the latter means that $N' = \alpha^-$, and then $\mathtt{nf}\,(P') = \exists \alpha^-.\downarrow \alpha^-$ for some α^-. Finally, notice that all the types of shape $\exists \alpha^-.\downarrow \alpha^-$ are equal.

■

Lemma 97 (Completeness of Entry Singularity)**.**

– *Suppose that there exists N well-formed in Θ such that for any N' well-formed in Θ, $\Theta \vdash N' : e$ implies $\Theta \vdash N' \simeq^{\leqslant} N$. Then e $\mathtt{singular\ with\ nf}\,(N)$.*

\+ *Suppose that there exists P well-formed in Θ such that for any P' well-formed in Θ, $\Theta \vdash P' : e$ implies $\Theta \vdash P' \simeq^{\leqslant} P$. Then e $\mathtt{singular\ with\ nf}\,(P)$.*

Proof

– By Lemma 84, there exists $\Theta \vdash N' : e$. Since N' is negative, by inversion of $\Theta \vdash N' : e$, e has shape $\widehat{\alpha^-} :\simeq M$, where $\Theta \vdash N' \simeq^{\leqslant} M$, and transitively, $\Theta \vdash N \simeq^{\leqslant} M$. Then $\mathtt{nf}\,(M) = \mathtt{nf}\,(N)$, and e $\mathtt{singular\ with\ nf}\,(M)$ (by $(\simeq^{\text{SING}}_{-})$) is rewritten as e $\mathtt{singular\ with\ nf}\,(N)$.

\+ By Lemma 84, there exists $\Theta \vdash P' : e$, then by assumption, $\Theta \vdash P' \simeq^{\leqslant} P$, which by Lemma 85 implies $\Theta \vdash P : e$.

Let us consider the shape of e:

Case 1. $e = (\widehat{\alpha}^{+} :\simeq Q)$ then inversion of $\Theta \vdash P : e$ implies $\Theta \vdash P \simeq^{\leqslant} Q$, and hence, $\mathtt{nf}\,(P) = \mathtt{nf}\,(Q)$ (by Lemma 48). Then $e\ \mathtt{singular\ with\ nf}\,(Q)$, which holds by $(\simeq^{\text{SING}}_{+})$, is rewritten as $e\ \mathtt{singular\ with\ nf}\,(P)$.

Case 2. $e = (\widehat{\alpha}^{+} :\geqslant Q)$. Then the inversion of $\Theta \vdash P : e$ implies $\Theta \vdash P \geqslant Q$. Let us consider the shape of Q:

a. $Q = \exists\overrightarrow{\beta^{-}}.\,\beta^{+}$ (for potentially empty $\overrightarrow{\beta^{-}}$). Then $\Theta \vdash P \geqslant \exists\overrightarrow{\beta^{-}}.\,\beta^{+}$ implies $\Theta \vdash P \simeq^{\leqslant} \beta^{+}$ by Lemma 77, as was noted in the proof of Lemma 96, and hence, $\mathtt{nf}\,(P) = \beta^{+}$.

Then $e\ \mathtt{singular\ with}\ \beta^{+}$, which holds by $(:\geqslant\alpha^{\text{SING}})$, can be rewritten as $e\ \mathtt{singular\ with\ nf}\,(P)$.

b. $Q = \exists\overrightarrow{\beta^{-}}.\downarrow N$ (for potentially empty $\overrightarrow{\beta^{-}}$). Notice that $\Theta \vdash \exists\gamma^{-}.\downarrow\gamma^{-} \geqslant \exists\overrightarrow{\beta^{-}}.\downarrow N$ (by $(\exists^{\geqslant})$, with substitution N/γ^{-}), and thus, $\Theta \vdash \exists\gamma^{-}.\downarrow\gamma^{-} : e$ by $(:\geqslant^{\text{SAT}}_{+})$.

Then by assumption, $\Theta \vdash \exists\gamma^{-}.\downarrow\gamma^{-} \simeq^{\leqslant} P$, that is $\mathtt{nf}\,(P) = \exists\gamma^{-}.\downarrow\gamma^{-}$. To apply $(:\geqslant\downarrow^{\text{SING}})$ to infer $(\widehat{\alpha}^{+} :\geqslant \exists\overrightarrow{\beta^{-}}.\downarrow N)\ \mathtt{singular\ with}\ \exists\gamma^{-}.\downarrow\gamma^{-}$, it is left to show that $N \simeq^{D} \beta^{-}{}_{i}$ for some i.

Since $\Theta \vdash Q : e$, by assumption, $\Theta \vdash Q \simeq^{\leqslant} P$, and by transitivity, $\Theta \vdash Q \simeq^{\leqslant} \exists\gamma^{-}.\downarrow\gamma^{-}$. It implies $\mathtt{nf}\,(\exists\overrightarrow{\beta^{-}}.\downarrow N) = \exists\gamma^{-}.\downarrow\gamma^{-}$ (by Lemma 48), which by definition of normalization means $\exists\overrightarrow{\beta^{-}}'.\downarrow\mathtt{nf}\,(N) = \exists\gamma^{-}.\downarrow\gamma^{-}$, where $\mathtt{ord}\,\overrightarrow{\beta^{-}}\ \mathtt{in}\ N' = \overrightarrow{\beta^{-}}'$. This way, $\overrightarrow{\beta^{-}}'$ is a variable β^{-}, and $\mathtt{nf}\,(N) = \beta^{-}$. Notice that $\beta^{-} \in \overrightarrow{\beta^{-}}' \subseteq \overrightarrow{\beta^{-}}$ by Lemma 35. This way, $N \simeq^{D} \beta^{-}$ for $\beta^{-} \in \overrightarrow{\beta^{-}}$ (by Lemma 48),

■

Lemma 98 (Soundness of Singularity)**.** *Suppose* $\Xi \vdash C : \widehat{\Theta}$*, and* $C\ \mathtt{singular\ with}\ \widehat{\sigma}$*. Then* $\Xi \vdash \widehat{\sigma} : \widehat{\Theta}$*,* $\Xi \vdash \widehat{\sigma} : C$*,* $\widehat{\sigma}$ *is normalized, and for any* $\widehat{\sigma}'$ *such that* $\Xi \vdash \widehat{\sigma} : C$*,* $\Xi \vdash \widehat{\sigma}' \simeq^{\leqslant} \widehat{\sigma} : \widehat{\Theta}$*.*

Proof Suppose that $\Xi \vdash \widehat{\sigma}' : C$. It means that for every $e \in C$ restricting $\widehat{\alpha}^{\pm}$, $\Xi(\widehat{\alpha}^{\pm}) \vdash [\widehat{\sigma}']\widehat{\alpha}^{\pm} : e$ holds. $C\ \mathtt{singular\ with}\ \widehat{\sigma}$ means $e\ \mathtt{singular\ with}\ [\widehat{\sigma}]\widehat{\alpha}^{\pm}$, and hence, by Lemma 97, $\Xi(\widehat{\alpha}^{\pm}) \vdash [\widehat{\sigma}']\widehat{\alpha}^{\pm} \simeq^{\leqslant} [\widehat{\sigma}]\widehat{\alpha}^{\pm}$ holds.

Since the uniqueness holds for every variable from $\mathtt{dom}\,(C)$, $\widehat{\sigma}$ is equivalent to $\widehat{\sigma}'$ on this set. ■

Observation 18 (Singularity is Deterministic)**.** *For a fixed* C *such that* $\Xi \vdash C : \widehat{\Theta}$*, if* $C\ \mathtt{singular\ with}\ \widehat{\sigma}$ *and* $C\ \mathtt{singular\ with}\ \widehat{\sigma}'$*, then* $\widehat{\sigma} = \widehat{\sigma}'$*.*

Proof By Lemma 98, $\Xi \vdash \widehat{\sigma} : \widehat{\Theta}$ and $\Xi \vdash \widehat{\sigma}' : \widehat{\Theta}$. It means that both $\widehat{\sigma}$ and $\widehat{\sigma}'$ act as identity outside of $\widehat{\Theta}$.

Moreover, for any $\widehat{\alpha}^{\pm} \in \widehat{\Theta}$, $\Xi \vdash C : \widehat{\Theta}$ means that there is a unique $e \in C$ restricting $\widehat{\alpha}^{\pm}$. Then $C\ \mathtt{singular\ with}\ \widehat{\sigma}$ means that $e\ \mathtt{singular\ with}\ [\widehat{\sigma}]\widehat{\alpha}^{\pm}$. By looking at the inference

rules, it is easy to see that $[\widehat{\sigma}]\widehat{\alpha}^{\pm}$ is uniquely determined by e, which, Similarly, $[\widehat{\sigma}']\widehat{\alpha}^{\pm}$ is also uniquely determined by e, in the same way, and hence, $[\widehat{\sigma}]\widehat{\alpha}^{\pm} = [\widehat{\sigma}']\widehat{\alpha}^{\pm}$. ■

Lemma 99 (Completeness of Singularity)**.** *For a given $\Xi \vdash C$, suppose that all the substitutions satisfying C are equivalent on $\widehat{\Theta} \supseteq \mathtt{dom}\,(C)$. In other words, suppose that there exists $\Xi \vdash \widehat{\sigma}_1 : \widehat{\Theta}$ such that for any $\Xi \vdash \widehat{\sigma} : \widehat{\Theta}$, $\Xi \vdash \widehat{\sigma} : C$ implies $\Xi \vdash \widehat{\sigma} \simeq^{\leqslant} \widehat{\sigma}_1 : \widehat{\Theta}$. Then*

- *C $\mathtt{singular\ with}$ $\widehat{\sigma}_0$ for some $\widehat{\sigma}_0$ and*
- *$\widehat{\Theta} = \mathtt{dom}\,(C)$.*

Proof

First, let us assume $\widehat{\Theta} \neq \mathtt{dom}\,(C)$. Then there exists $\widehat{\alpha}^{\pm} \in \widehat{\Theta} \setminus \mathtt{dom}\,(C)$. Let us take $\Xi \vdash \widehat{\sigma}_1 : \widehat{\Theta}$ such that any other substitution $\Xi \vdash \widehat{\sigma} : \widehat{\Theta}$ satisfying C is equivalent to $\widehat{\sigma}_1$ on $\widehat{\Theta}$.

Notice that $\Xi \vdash \widehat{\sigma}_1 : C$: by Lemma 84, there exists $\widehat{\sigma}'$ such that $\Xi \vdash \widehat{\sigma}' : \widehat{\Theta}$ and $\Xi \vdash \widehat{\sigma}' : C$, and by assumption, $\Xi \vdash \widehat{\sigma}' \simeq^{\leqslant} \widehat{\sigma}_1 : \widehat{\Theta}$, implying $\Xi \vdash \widehat{\sigma}' \simeq^{\leqslant} \widehat{\sigma}_1 : \mathtt{dom}\,(C)$.

Let us construct $\widehat{\sigma}_2$ such that $\Xi \vdash \widehat{\sigma}_2 : \widehat{\Theta}$ as follows:

$$\begin{cases} [\widehat{\sigma}_2]\widehat{\beta}^{\pm} = [\widehat{\sigma}_1]\widehat{\beta}^{\pm} & \text{if } \widehat{\beta}^{\pm} \neq \widehat{\alpha}^{\pm} \\ [\widehat{\sigma}_2]\widehat{\alpha}^{\pm} = T & \text{where } T \text{ is any closed type not equivalent to } [\widehat{\sigma}_1]\widehat{\alpha}^{\pm} \end{cases}$$

It is easy to see that $\Xi \vdash \widehat{\sigma}_2 : C$ since $\widehat{\sigma}_1|_{\mathtt{dom}\,(C)} = \widehat{\sigma}_2|_{\mathtt{dom}\,(C)}$, and $\Xi \vdash \widehat{\sigma}_1 : C$. However, $\Xi \vdash \widehat{\sigma}_2 \simeq^{\leqslant} \widehat{\sigma}_1 : \widehat{\Theta}$ does not hold because by construction, $\Xi(\widehat{\alpha}^{\pm}) \vdash [\widehat{\sigma}_2]\widehat{\alpha}^{\pm} \simeq^{\leqslant} [\widehat{\sigma}_1]\widehat{\alpha}^{\pm}$ does not hold. This way, we have a contradiction.

Second, let us show C $\mathtt{singular\ with}$ $\widehat{\sigma}_0$. Let us take arbitrary $e \in C$ restricting $\widehat{\beta}^{\pm}$. We need to show that e is singular. Notice that $\Xi \vdash \widehat{\sigma}_1 : C$ implies $\Xi(\widehat{\beta}^{\pm}) \vdash [\widehat{\sigma}_1]\widehat{\beta}^{\pm}$ and $\Xi(\widehat{\beta}^{\pm}) \vdash [\widehat{\sigma}_1]\widehat{\beta}^{\pm} : e$. We will show that any other type satisfying e is equivalent to $[\widehat{\sigma}_1]\widehat{\beta}^{\pm}$, then by Lemma 97, e $\mathtt{singular\ with}$ $[\widehat{\sigma}_1]\widehat{\beta}^{\pm}$.

- if $\widehat{\beta}^{\pm}$ is positive, let us take any type $\Xi(\widehat{\beta}^{\pm}) \vdash P'$ and assume $\Xi(\widehat{\beta}^{\pm}) \vdash P' : e$. We will show that $\Xi(\widehat{\beta}^{\pm}) \vdash P' \simeq^{\leqslant} [\widehat{\sigma}_1]\widehat{\beta}^{\pm}$, which by Lemma 48 will imply e $\mathtt{singular\ with\ nf}$ $([\widehat{\sigma}_1]\widehat{\beta}^{\pm})$.

  Let us construct $\widehat{\sigma}_2$ such that $\Xi \vdash \widehat{\sigma}_2 : \widehat{\Theta}$ as follows:

  $$\begin{cases} [\widehat{\sigma}_2]\widehat{\gamma}^{\pm} = [\widehat{\sigma}_1]\widehat{\gamma}^{\pm} & \text{if } \widehat{\gamma}^{\pm} \neq \widehat{\beta}^{\pm} \\ [\widehat{\sigma}_2]\widehat{\beta}^{\pm} = P' \end{cases}$$

  It is easy to see that $\Xi \vdash \widehat{\sigma}_2 : C$: for e, $\Xi(\widehat{\beta}^{\pm}) \vdash [\widehat{\sigma}_2]\widehat{\beta}^{\pm} : e$ by construction, since $\Xi(\widehat{\beta}^{\pm}) \vdash P' : e$; for any other $e' \in C$ restricting $\widehat{\gamma}^{\pm}$, $[\widehat{\sigma}_2]\widehat{\gamma}^{\pm} = [\widehat{\sigma}_1]\widehat{\gamma}^{\pm}$, and $\Xi(\widehat{\gamma}^{\pm}) \vdash [\widehat{\sigma}_1]\widehat{\gamma}^{\pm} : e'$ since $\Xi \vdash \widehat{\sigma}_1 : C$.

  Then by assumption, $\Xi \vdash \widehat{\sigma}_2 \simeq^{\leqslant} \widehat{\sigma}_1 : \widehat{\Theta}$, which in particular means $\Xi(\widehat{\beta}^{\pm}) \vdash [\widehat{\sigma}_2]\widehat{\beta}^{\pm} \simeq^{\leqslant} [\widehat{\sigma}_1]\widehat{\beta}^{\pm}$, that is $\Xi(\widehat{\beta}^{\pm}) \vdash P' \simeq^{\leqslant} [\widehat{\sigma}_1]\widehat{\beta}^{\pm}$.

- if $\widehat{\beta}^{\pm}$ is negative, the proof is analogous.

■

C.5.2 Correctness of the Typing Algorithm

Lemma 100 (Determinacy of typing algorithm)**.** *Suppose that* $\Theta \vdash \Gamma$ *and* $\Theta \vdash^{\supseteq} \Xi$*. Then*

- + *If* $\Theta; \Gamma \vDash v\colon P$ *and* $\Theta; \Gamma \vDash v\colon P'$ *then* $P = P'$*.*
- − *If* $\Theta; \Gamma \vDash c\colon N$ *and* $\Theta; \Gamma \vDash c\colon N'$ *then* $N = N'$*.*
- • *If* $\Theta\,;\Gamma\,;\Xi \vDash N \bullet \vec{v} \Rrightarrow M \dashv \Xi'\,;C$ *and* $\Theta\,;\Gamma\,;\Xi \vDash N \bullet \vec{v} \Rrightarrow M' \dashv \Xi'\,;C'$ *then* $M = M'$*,* $\Xi = \Xi'$*, and* $C = C'$*.*

Proof We show it by structural induction on the inference tree. Notice that the last rule used to infer the judgement is uniquely determined by the input, and that each premise of each inference rule is deterministic by the corresponding observation. ■

Let us extend the declarative typing metric (Definition 33) to the algorithmic typing.

Definition 34 (Size of an Algorithmic Judgement)**.** *For an algorithmic typing judgement* J *let us define a metrics* $\mathsf{size}(J)$ *as a pair of numbers in the following way:*

- + $\mathsf{size}(\Theta; \Gamma \vDash v\colon P) = (\mathsf{size}(v), 0)$*;*
- − $\mathsf{size}(\Theta; \Gamma \vDash c\colon N) = (\mathsf{size}(c), 0)$*;*
- • $\mathsf{size}(\Theta\,;\Gamma\,;\Xi \vDash N \bullet \vec{v} \Rrightarrow M \dashv \Xi'\,;C) = (\mathsf{size}(\vec{v}), \mathsf{npq}(N)))$

Definition 35 (Metric)**.** *We extend the metric from Definition 33 to the algorithmic typing in the following way. For a tree* T *inferring an algorithmic typing judgement* J*, we define* $\mathsf{metric}(T)$ *as* $(\mathsf{size}(J), 0)$*.*

Soundness and completeness are proved by mutual induction on the metric of the inference tree.

Lemma 101 (Soundness of typing)**.** *Suppose that* $\Theta \vdash \Gamma$*. For an inference tree* T_1*,*

- + *If* T_1 *infers* $\Theta; \Gamma \vDash v\colon P$ *then* $\Theta \vdash P$ *and* $\Theta; \Gamma \vdash v\colon P$
- − *If* T_1 *infers* $\Theta; \Gamma \vDash c\colon N$ *then* $\Theta \vdash N$ *and* $\Theta; \Gamma \vdash c\colon N$
- • *If* T_1 *infers* $\Theta\,;\Gamma\,;\Xi \vDash N \bullet \vec{v} \Rrightarrow M \dashv \Xi'\,;C$ *for* $\Theta \vdash^{\supseteq} \Xi$ *and* $\Theta\,;\mathtt{dom}\,(\Xi) \vdash N$ *free from negative algorithmic variables, then*
  1. $\Theta \vdash^{\supseteq} \Xi'$
  2. $\Xi \subseteq \Xi'$
  3. $\Theta\,;\mathtt{dom}\,(\Xi') \vdash M$
  4. $\mathtt{dom}\,(\Xi) \cap \mathtt{fav}(M) \subseteq \mathtt{fav} N$
  5. M *is normalized and free from negative algorithmic variables*
  6. $\Xi'|_{\mathtt{fav} N \cup \mathtt{fav} M} \vdash C$
  7. *for any* $\Xi' \vdash \widehat{\sigma} : \mathtt{fav} N \cup \mathtt{fav} M, \Xi' \vdash \widehat{\sigma} : C$ *implies* $\Theta\,;\Gamma \vdash [\widehat{\sigma}]N \bullet \vec{v} \Rrightarrow [\widehat{\sigma}]M$

Proof We prove it by induction on $\mathsf{metric}(T_1)$, mutually with the completeness of typing (Lemma 101). Let us consider the last rule used to infer the derivation.

Case 1. ($\textsc{Var}^{\textsc{inf}}$) We are proving that if $\Theta; \Gamma \vDash x \colon \mathtt{nf}\,(P)$ then $\Theta \vdash \mathtt{nf}\,(P)$ and $\Theta; \Gamma \vdash x \colon \mathtt{nf}\,(P)$.

By inversion, $x : P \in \Gamma$. Since $\Theta \vdash \Gamma$, we have $\Theta \vdash P$, and by Corollary 14, $\Theta \vdash \mathtt{nf}\,(P)$.

By applying ($\textsc{Var}^{\textsc{inf}}$) to $x : P \in \Gamma$, we infer $\Theta; \Gamma \vdash x \colon P$. Finally, by ($\simeq_+^{\textsc{inf}}$), since $\Theta \vdash P \simeq^{\leqslant} \mathtt{nf}\,(P)$ (Corollary 16), we have $\Theta; \Gamma \vdash x \colon \mathtt{nf}\,(P)$.

Case 2. ($\{\}^{\textsc{inf}}$)

We are proving that if $\Theta; \Gamma \vDash \{c\} \colon \downarrow N$ then $\Theta \vdash \downarrow N$ and $\Theta; \Gamma \vdash \{c\} \colon \downarrow N$.

By inversion of $\Theta; \Gamma \vDash \{c\} \colon \downarrow N$, we have $\Theta; \Gamma \vDash c \colon N$. By the induction hypothesis applied to $\Theta; \Gamma \vDash c \colon N$, we have

1. $\Theta \vdash N$, and hence, $\Theta \vdash \downarrow N$;
2. $\Theta; \Gamma \vdash c \colon N$, which by ($\{\}^{\textsc{inf}}$) implies $\Theta; \Gamma \vdash \{c\} \colon \downarrow N$.

Case 3. ($\textsc{ret}^{\textsc{inf}}$) The proof is symmetric to the previous case (case 2).

Case 4. ($\textsc{ann}_+^{\textsc{inf}}$) We are proving that if $\Theta; \Gamma \vDash (v : Q) \colon \mathtt{nf}\,(Q)$ then $\Theta \vdash \mathtt{nf}\,(Q)$ and $\Theta; \Gamma \vdash (v : Q) \colon \mathtt{nf}\,(Q)$.

By inversion of $\Theta; \Gamma \vDash (v : Q) \colon \mathtt{nf}\,(Q)$, we have:

1. $\Theta \vdash (v : Q)$, hence, $\Theta \vdash Q$, and by Corollary 14, $\Theta \vdash \mathtt{nf}\,(Q)$;
2. $\Theta; \Gamma \vDash v \colon P$, which by the induction hypothesis implies $\Theta \vdash P$ and $\Theta; \Gamma \vdash v \colon P$;
3. $\Theta; \cdot \vDash Q \geqslant P \dashv \cdot$, which by Lemma 86 implies $\Theta \vdash [\cdot]\,Q \geqslant P$, that is $\Theta \vdash Q \geqslant P$.

To infer $\Theta; \Gamma \vdash (v : Q) \colon Q$, we apply ($\textsc{ann}_+^{\textsc{inf}}$) to $\Theta; \Gamma \vdash v \colon P$ and $\Theta \vdash Q \geqslant P$. Then by ($\simeq_+^{\textsc{inf}}$), $\Theta; \Gamma \vdash (v : Q) \colon \mathtt{nf}\,(Q)$.

Case 5. ($\textsc{ann}_-^{\textsc{inf}}$) The proof is symmetric to the previous case (case 4).

Case 6. ($\lambda^{\textsc{inf}}$) We are proving that if $\Theta; \Gamma \vDash \lambda x : P.\, c \colon \mathtt{nf}\,(P \to N)$ then $\Theta \vdash \mathtt{nf}\,(P \to N)$ and $\Theta; \Gamma \vdash \lambda x : P.\, c \colon \mathtt{nf}\,(P \to N)$.

By inversion of $\Theta; \Gamma \vDash \lambda x : P.\, c \colon \mathtt{nf}\,(P \to N)$, we have $\Theta \vdash \lambda x : P.\, c$, which implies $\Theta \vdash P$.

Also by inversion of $\Theta; \Gamma \vDash \lambda x : P.\, c \colon \mathtt{nf}\,(P \to N)$, we have $\Theta; \Gamma, x : P \vDash c \colon N$, applying induction hypothesis to which gives us:

1. $\Theta \vdash N$, thus $\Theta \vdash P \to N$, and by Corollary 14, $\Theta \vdash \mathtt{nf}\,(P \to N)$;
2. $\Theta; \Gamma, x : P \vdash c \colon N$, which by ($\lambda^{\textsc{inf}}$) implies $\Theta; \Gamma \vdash \lambda x : P.\, c \colon P \to N$, and by ($\simeq_+^{\textsc{inf}}$), $\Theta; \Gamma \vdash \lambda x : P.\, c \colon \mathtt{nf}\,(P \to N)$.

Case 7. ($\Lambda^{\textsc{inf}}$) We are proving that if $\Theta; \Gamma \vDash \Lambda \alpha^+.\, c \colon \mathtt{nf}\,(\forall \alpha^+.\, N)$ then $\Theta; \Gamma \vdash \Lambda \alpha^+.\, c \colon \mathtt{nf}\,(\forall \alpha^+.\, N)$ and $\Theta \vdash \mathtt{nf}\,(\forall \alpha^+.\, N)$.

By inversion of $\Theta, \alpha^+; \Gamma \vDash c \colon N$, we have $\Theta \vdash \Lambda \alpha^+.\, c$, which implies $\Theta, \alpha^+ \vdash c$.

Also by inversion of $\Theta, \alpha^+; \Gamma \vDash c \colon N$, we have $\Theta, \alpha^+; \Gamma \vDash c \colon N$. Obtaining the induction hypothesis to $\Theta, \alpha^+; \Gamma \vDash c \colon N$, we have:

1. $\Theta, \alpha^+ \vdash N$, thus $\Theta \vdash \forall \alpha^+.\, N$, and by Corollary 14, $\Theta \vdash \mathtt{nf}\ (\forall \alpha^+.\, N)$;
2. $\Theta, \alpha^+; \Gamma \vdash c : N$, which by (Λ^{INF}) implies $\Theta; \Gamma \vdash \Lambda \alpha^+.\, c : \forall \alpha^+.\, N$, and by $(\simeq_+^{\mathrm{INF}})$, $\Theta; \Gamma \vdash \Lambda \alpha^+.\, c : \mathtt{nf}\ (\forall \alpha^+.\, N)$.

Case 8. $(\mathrm{LET}^{\mathrm{INF}})$ We are proving that if $\Theta; \Gamma \vDash \mathtt{let}\, x = v;\ c : N$ then $\Theta; \Gamma \vdash \mathtt{let}\, x = v;\ c : N$ and $\Theta \vdash N$.

By inversion of $\Theta; \Gamma \vDash \mathtt{let}\, x = v;\ c : N$, we have:

1. $\Theta; \Gamma \vDash v : P$, which by the induction hypothesis implies $\Theta \vdash P$ (and thus, $\Theta \vdash \Gamma, x : P$) and $\Theta; \Gamma \vdash v : P$;
2. $\Theta; \Gamma, x : P \vDash c : N$, which by the induction hypothesis implies $\Theta \vdash N$ and $\Theta; \Gamma, x : P \vdash c : N$.

This way, $\Theta; \Gamma \vdash \mathtt{let}\, x = v;\ c : N$ holds by $(\mathrm{LET}^{\mathrm{INF}})$.

Case 9. $(\mathrm{LET}_{\mathrm{C}}^{\mathrm{INF}})$ We are proving that if $\Theta; \Gamma \vDash \mathtt{let}\, x : P = c;\ c' : N$ then $\Theta; \Gamma \vdash \mathtt{let}\, x : P = c;\ c' : N$ and $\Theta \vdash N$.

By inversion of $\Theta; \Gamma \vDash \mathtt{let}\, x : P = c;\ c' : N$, we have:

1. $\Theta; \Gamma \vDash c : M$, which by the induction hypothesis implies $\Theta \vdash M$ and $\Theta; \Gamma \vdash c : M$;
2. $\Theta;\ \cdot \vDash M \leqslant \uparrow P \dashv \cdot$, which by the soundness of negative subtyping (Lemma 92) means $\Theta \vdash M \leqslant \uparrow P$;
3. $\Theta; \Gamma, x : P \vDash c' : N$, which by the induction hypothesis implies $\Theta \vdash N$ and $\Theta; \Gamma, x : P \vdash c' : N$.

This way, $\Theta; \Gamma \vdash \mathtt{let}\, x : P = c;\ c' : N$ holds by $(\mathrm{LET}_{\mathrm{C}}^{\mathrm{INF}})$.

Case 10. $(\mathrm{LET}_{:@}^{\mathrm{INF}})$ We are proving that if $\Theta; \Gamma \vDash \mathtt{let}\, x : P = v(\vec{v});\ c' : N$ then $\Theta; \Gamma \vdash \mathtt{let}\, x : P = v(\vec{v});\ c' : N$ and $\Theta \vdash N$.

By inversion, we have:

1. $\Theta \vdash P$, hence, $\Theta \vdash \Gamma, x : P$
2. $\Theta; \Gamma \vDash v : \downarrow M$
3. $\Theta\,;\Gamma\,;\cdot \vDash M \bullet \vec{v} \Longrightarrow M' \dashv \Xi\,; C_1$
4. $\Theta;\ \Xi \vDash M' \leqslant \uparrow P \dashv C_2$
5. $\Xi \vdash C_1\ \&\ C_2 = C$
6. $\Theta; \Gamma, x : P \vDash c' : N$

By the induction hypothesis applied to $\Theta; \Gamma \vDash v : \downarrow M$, we have $\Theta; \Gamma \vdash v : \downarrow M$ and $\Theta \vdash \downarrow M$ (and hence, $\Theta\,;\mathtt{dom}\,(\Xi) \vdash M$).

By the induction hypothesis applied to $\Theta; \Gamma, x : P \vDash c' : N$, we have $\Theta; \Gamma, x : P \vdash c' : N$ and $\Theta \vdash N$.

By the induction hypothesis applied to $\Theta\,;\Gamma\,;\cdot \vDash M \bullet \vec{v} \Longrightarrow M' \dashv \Xi\,; C_1$, we have:

1. $\Theta \vdash^{\supseteq} \Xi$,
2. $\Theta\,;\mathtt{dom}\,(\Xi) \vdash M'$,
3. $\Xi'|_{\mathtt{fav} M \cup \mathtt{fav} Q} \vdash C_1$, and thus, $\mathtt{dom}\,(C_1) \subseteq \mathtt{fav} M \cup \mathtt{fav} M'$.
4. for any $\Xi' \vdash \widehat{\sigma} : C_1$, we have $\Theta\,;\Gamma \vdash [\widehat{\sigma}]M \bullet \vec{v} \Rightarrow [\widehat{\sigma}]M'$.

By soundness of negative subtyping (Lemma 92) applied to $\Theta;\ \Xi \vDash M' \leqslant \uparrow P \dashv C_2$, we have $\Xi \vdash C_2 : \mathtt{fav}(M')$, and thus, $\mathtt{fav}(M') = \mathtt{dom}\,(C_2)$.

By soundness of constraint merge (Lemma 89), $\mathtt{dom}\,(C) = \mathtt{dom}\,(C_1) \cup \mathtt{dom}\,(C_2) \subseteq \mathtt{fav} M \cup \mathtt{fav} M'$ Then by Lemma 84, let us take $\widehat{\sigma}$ such that $\Xi \vdash \widehat{\sigma} : \mathtt{fav}(M) \cup \mathtt{fav}(M')$ and $\Xi \vdash \widehat{\sigma} : C$. By the soundness of constraint merge, $\Xi \vdash \widehat{\sigma} : C_1$ and $\Xi \vdash \widehat{\sigma} : C_2$, and by weakening, $\Xi' \vdash \widehat{\sigma} : C_1$ and $\Xi' \vdash \widehat{\sigma} : C_2$.

Then as noted above (4), $\Theta\,;\Gamma \vdash M \bullet \vec{v} \Rightarrow [\widehat{\sigma}]M'$ And again, by the soundness of negative subtyping (Lemma 92) applied to $\Theta;\ \Xi \vDash M' \leqslant \uparrow P \dashv C_2$, we have $\Theta \vdash [\widehat{\sigma}]M' \leqslant \uparrow P$.

To infer $\Theta;\Gamma \vdash \mathtt{let}\, x : P = v(\vec{v});\ c' : N$, we apply the corresponding declarative rule $(\textsc{let}^{\textsc{inf}}_{:@})$, where Q is $[\widehat{\sigma}]Q$. Notice that all the premises were already shown to hold above:

1. $\Theta \vdash P$ and $\Theta;\Gamma \vdash v : \downarrow M$ from the assumption,
2. $\Theta\,;\Gamma \vdash M \bullet \vec{v} \Rightarrow [\widehat{\sigma}]M'$ holds as noted above,
3. $\Theta \vdash [\widehat{\sigma}]M' \leqslant \uparrow P$ by soundness of negative subtyping,
4. $\Theta;\Gamma, x : P \vdash c' : N$ from the the induction hypothesis.

Case 11. $(\textsc{let}^{\textsc{inf}}_{@})$ We are proving that if $\Theta;\Gamma \vDash \mathtt{let}\, x = v(\vec{v});\ c' : N$ then $\Theta;\Gamma \vdash \mathtt{let}\, x = v(\vec{v});\ c' : N$ and $\Theta \vdash N$.

By the inversion, we have:

1. $\Theta;\Gamma \vDash v : \downarrow M$,
2. $\Theta\,;\Gamma\,;\cdot \vDash M \bullet \vec{v} \Rightarrow \uparrow Q \dashv \Xi\,;C$,
3. Q is C-minimized by $\widehat{\sigma}$, and
4. $\Theta;\Gamma, x : [\widehat{\sigma}]Q \vDash c' : N$.

By the induction hypothesis applied to $\Theta;\Gamma \vDash v : \downarrow M$, we have $\Theta;\Gamma \vdash v : \downarrow M$ and $\Theta \vdash \downarrow M$ (and thus, $\Theta\,;\mathtt{dom}\,(\Xi) \vdash M$).

By the induction hypothesis applied to $\Theta;\Gamma, x : [\widehat{\sigma}]Q \vDash c' : N$, we have $\Theta \vdash N$ and $\Theta;\Gamma, x : [\widehat{\sigma}]Q \vdash c' : N$.

By the induction hypothesis applied to $\Theta\,;\Gamma\,;\cdot \vDash M \bullet \vec{v} \Rightarrow \uparrow Q \dashv \Xi\,;C$, we have:

1. $\Theta \vdash^{\supseteq} \Xi$
2. $\Theta\,;\mathtt{dom}\,(\Xi) \vdash \uparrow Q$
3. $\Xi|_{\mathtt{fav} M \cup \mathtt{fav} Q} \vdash C$ (and thus, $\mathtt{dom}\,(C) \subseteq \mathtt{fav} M \cup \mathtt{fav} Q$)

4. for any $\Xi \vdash \widehat{\sigma} : C$, we have $\Theta ; \Gamma \vdash [\widehat{\sigma}] M \bullet \vec{v} \Longrightarrow [\widehat{\sigma}] \uparrow Q$, which, since M is ground means $\Theta ; \Gamma \vdash M \bullet \vec{v} \Longrightarrow \uparrow [\widehat{\sigma}] Q$.

To infer $\Theta ; \Gamma \vdash \mathtt{let}\, x = v(\vec{v}) ;\ c' : N$, we apply the corresponding declarative rule ($\mathrm{LET}^{\mathrm{INF}}_{@}$). Let us show that the premises hold:

- $\Theta ; \Gamma \vdash v : \downarrow M$ holds by the induction hypothesis;
- $\Theta ; \Gamma, x : [\widehat{\sigma}] Q \vdash c' : N$ also holds by the induction hypothesis, as noted above;
- $\Theta ; \Gamma \vdash M \bullet \vec{v} \Longrightarrow \uparrow [\widehat{\sigma}] Q$ holds, as noted above;
- To show the principality of $\uparrow [\widehat{\sigma}] Q$, we assume that for some other type R holds $\Theta ; \Gamma \vdash M \bullet \vec{v} \Longrightarrow \uparrow R$, that is $\Theta ; \Gamma \vdash [\cdot] M \bullet \vec{v} \Longrightarrow \uparrow R$. Then by the completeness of typing (Lemma 102), there exist N', Ξ', and C' such that
  1. $\Theta ; \Gamma ; \cdot \vDash M \bullet \vec{v} \Longrightarrow N' \dashv \Xi' ; C'$ and
  2. there exists a substitution $\Xi' \vdash \widehat{\sigma}' : C'$ such that $\Theta \vdash [\widehat{\sigma}'] N' \simeq^{\leqslant} \uparrow R$.

  By determinacy of the typing algorithm (Lemma 100), $\Theta ; \Gamma ; \cdot \vDash M \bullet \vec{v} \Longrightarrow > N' \dashv \Xi' ; C'$, means that C' is C, Ξ' is Ξ, and N' is $\uparrow Q$. This way, $\Theta \vdash [\widehat{\sigma}'] \uparrow Q \simeq^{\leqslant} \uparrow R$ for substitution $\Xi \vdash \widehat{\sigma}' : C$. To show the principality, it suffices to notice that $\Theta \vdash R \geqslant [\widehat{\sigma}] Q$ or equivalently $\Theta \vdash [\widehat{\sigma}'] Q \geqslant [\widehat{\sigma}] Q$, which holds by the soundness of the minimal instantiation (Lemma 94) since Q is C-minimized by $\widehat{\sigma}$.

Case 12. ($\mathrm{LET}^{\mathrm{INF}}_{\exists}$) We are proving that if $\Theta ; \Gamma \vDash \mathtt{let}^{\exists}(\overrightarrow{\alpha}, x) = v ;\ c' : N$ then $\Theta ; \Gamma \vdash \mathtt{let}^{\exists}(\overrightarrow{\alpha}, x) = v ;\ c' : N$ and $\Theta \vdash N$. By the inversion, we have:

1. $\Theta ; \Gamma \vDash v : \exists \overrightarrow{\alpha}. P$
2. $\Theta, \overrightarrow{\alpha} ; \Gamma, x : P \vDash c' : N$
3. $\Theta \vdash N$

By the induction hypothesis applied to $\Theta ; \Gamma \vDash v : \exists \overrightarrow{\alpha}. P$, we have $\Theta ; \Gamma \vdash v : \exists \overrightarrow{\alpha}. P$ and $\exists \overrightarrow{\alpha}. P$ is normalized. By the induction hypothesis applied to $\Theta, \overrightarrow{\alpha} ; \Gamma, x : P \vDash c' : N$, we have $\Theta, \overrightarrow{\alpha} ; \Gamma, x : P \vdash c' : N$.

To show $\Theta ; \Gamma \vdash \mathtt{let}^{\exists}(\overrightarrow{\alpha}, x) = v ;\ c' : N$, we apply the corresponding declarative rule ($\mathrm{LET}^{\mathrm{INF}}_{\exists}$). Let us show that the premises hold:

1. $\Theta ; \Gamma \vdash v : \exists \overrightarrow{\alpha}. P$ holds by the induction hypothesis, as noted above,
2. $\mathtt{nf}\,(\exists \overrightarrow{\alpha}. P) = \exists \overrightarrow{\alpha}. P$ holds since $\exists \overrightarrow{\alpha}. P$ is normalized,
3. $\Theta, \overrightarrow{\alpha} ; \Gamma, x : P \vdash c' : N$ also holds by the induction hypothesis,
4. $\Theta \vdash N$ holds by the inversion, as noted above.

Case 13. ($\emptyset^{\mathrm{INF}}_{\bullet \Longrightarrow}$) Then by assumption:

- $\Theta \vdash^{\supseteq} \Xi$,
- $\Theta ; \mathtt{dom}\,(\Xi) \vdash N$ is free from negative algorithmic variables,

- $\Theta ; \Gamma ; \Xi \vDash N \bullet \Rightarrow\!\!> \mathtt{nf}(N) \dashv \Xi ; \cdot$.

Let us show the required properties:

1. $\Theta \vdash^{\supseteq} \Xi$ holds by assumption,
2. $\Xi \subseteq \Xi$ holds trivially,
3. $\mathtt{nf}(N)$ is evidently normalized, $\Theta ; \mathtt{dom}(\Xi) \vdash N$ implies $\Theta ; \mathtt{dom}(\Xi) \vdash \mathtt{nf}(N)$ by Corollary 24, and Lemma 40 means that $\mathtt{nf}(N)$ is inherently free from negative algorithmic variables,
4. $\mathtt{dom}(\Xi) \cap \mathtt{fav}(\mathtt{nf}(N)) \subseteq \mathtt{fav} N$ holds since $\mathtt{fav}(\mathtt{nf}(N)) = \mathtt{fav}(N)$,
5. $\Xi|_{\mathtt{fav} N \cup \mathtt{favnf}(N)} \vdash \cdot$ holds trivially,
6. suppose that $\Xi \vdash \widehat{\sigma} : \mathtt{fav} N \cup \mathtt{favnf}(N)$. To show $\Theta ; \Gamma \vdash [\widehat{\sigma}] N \bullet \Rightarrow\!\!> [\widehat{\sigma}] \mathtt{nf}(N)$, we apply the corresponding declarative rule $(\emptyset^{\mathrm{INF}}_{\bullet\Rightarrow})$. To show $\Theta \vdash [\widehat{\sigma}] N \simeq^{\leqslant} [\widehat{\sigma}] \mathtt{nf}(N)$, we apply the following sequence: $N \simeq^{D} \mathtt{nf}(N)$ by Lemma 41, then $[\widehat{\sigma}] N \simeq^{D} [\widehat{\sigma}] \mathtt{nf}(N)$ by Corollary 21, then $\Theta \vdash [\widehat{\sigma}] N \simeq^{\leqslant} [\widehat{\sigma}] \mathtt{nf}(N)$ by Lemma 29.

Case 14. $(\rightarrow^{\mathrm{INF}}_{\bullet\Rightarrow})$ By assumption:

1. $\Theta \vdash^{\supseteq} \Xi$,
2. $\Theta ; \mathtt{dom}(\Xi) \vdash Q \rightarrow N$ is free from negative algorithmic variables, and hence, so are Q and N,
3. $\Theta ; \Gamma ; \Xi \vDash Q \rightarrow N \bullet v, \vec{v} \Rightarrow\!\!> M \dashv \Xi' ; C$, and by inversion:
   a. $\Theta ; \Gamma \vDash v : P$, and by the induction hypothesis applied to this judgment, we have $\Theta ; \Gamma \vdash v : P$, and $\Theta \vdash P$;
   b. $\Theta ; \Xi \vDash Q \geqslant P \dashv C_1$, and by the soundness of subtyping: $\Xi \vdash C_1 : \mathtt{fav} Q$ (and thus, $\mathtt{dom}(C_1) = \mathtt{fav} Q$), and for any
      $\Xi \vdash \widehat{\sigma} : C_1$, we have $\Theta \vdash [\widehat{\sigma}] Q \geqslant P$;
   c. $\Theta ; \Gamma ; \Xi \vDash N \bullet \vec{v} \Rightarrow\!\!> M \dashv \Xi' ; C_2$, and by the induction hypothesis applied to this judgment,
      i. $\Theta \vdash^{\supseteq} \Xi'$,
      ii. $\Xi \subseteq \Xi'$,
      iii. $\Theta ; \mathtt{dom}(\Xi') \vdash M$ is normalized and free from negative algorithmic variables,
      iv. $\mathtt{dom}(\Xi) \cap \mathtt{fav}(M) \subseteq \mathtt{fav} N$,
      v. $\Xi'|_{\mathtt{fav}(M) \cup \mathtt{fav}(N)} \vdash C_2$, and thus, $\mathtt{dom}(C_2) \subseteq \mathtt{fav}(M) \cup \mathtt{fav}(N)$,
      vi. for any $\widehat{\sigma}$ such that $\Xi \vdash \widehat{\sigma} : \mathtt{fav}(M) \cup \mathtt{fav}(N)$ and $\Xi' \vdash \widehat{\sigma} : C_2$, we have $\Theta ; \Gamma \vdash [\widehat{\sigma}] N \bullet \vec{v} \Rightarrow\!\!> [\widehat{\sigma}] M$;

d. $\Xi \vdash C_1 \,\&\, C_2 = C$, which by Lemma 89 implies $\mathtt{dom}\,(C) = \mathtt{dom}\,(C_1) \cup \mathtt{dom}\,(C_2) \subseteq \mathtt{fav}\,Q \cup \mathtt{fav}\,M \cup \mathtt{fav}\,N$.

Let us show the required properties:

1. $\Theta \vdash^{\supseteq} \Xi'$ is shown above,
2. $\Xi \subseteq \Xi'$ is shown above,
3. $\Theta\,;\mathtt{dom}\,(\Xi') \vdash M$ is normalized and free from negative algorithmic variables, as shown above,
4. $\mathtt{dom}\,(\Xi) \cap \mathtt{fav}(M) \subseteq \mathtt{fav}\,N \subseteq \mathtt{fav}(Q \to N)$ (the first inclusion is shown above, the second one is by definition),
5. To show $\Xi'|_{\mathtt{fav}(Q)\cup\mathtt{fav}(N)\cup\mathtt{fav}(M)} \vdash C$, first let us notice that $\mathtt{fav}(Q) \cup \mathtt{fav}(N) \cup \mathtt{fav}(M) \subseteq \mathtt{dom}\,(C)$, as mentioned above. Then we demonstrate $\Xi' \vdash C$: $\Xi \vdash C_1$ and $\Xi \subseteq \Xi'$ imply $\Xi' \vdash C_1$, by the soundness of constraint merge (Lemma 89) applied to $\Xi' \vdash C_1 \,\&\, C_2 = C$:
   a. $\Xi' \vdash C$,
   b. for any $\Xi' \vdash \widehat{\sigma} : C$, $\Xi' \vdash \widehat{\sigma} : C_i$ holds;
6. Suppose that $\Xi' \vdash \widehat{\sigma} : \mathtt{fav}(Q) \cup \mathtt{fav}(N) \cup \mathtt{fav}(M)$ and $\Xi' \vdash \widehat{\sigma} : C$. To show $\Theta\,;\Gamma \vdash [\widehat{\sigma}](Q \to N) \bullet v, \vec{v} \Rrightarrow [\widehat{\sigma}]M$, that is $\Theta\,;\Gamma \vdash [\widehat{\sigma}]Q \to [\widehat{\sigma}]N \bullet v, \vec{v} \Rrightarrow [\widehat{\sigma}]M$, we apply the corresponding declarative rule $(\to^{\mathrm{INF}}_{\bullet\Rightarrow})$. Let us show the required premises:
   a. $\Theta; \Gamma \vdash v\colon P$ holds as shown above,
   b. $\Theta \vdash [\widehat{\sigma}]Q \geqslant P$ holds since $\Theta \vdash [\widehat{\sigma}|_{\mathtt{fav}(Q)}]Q \geqslant P$ by the soundness of subtyping as noted above: since $\Xi' \vdash \widehat{\sigma} : C$ implies $\Xi' \vdash \widehat{\sigma}|_{\mathtt{fav}(Q)} : C_1$, which we strengthen to $\Xi \vdash \widehat{\sigma}|_{\mathtt{fav}(Q)} : C_1$,
   c. $\Theta\,;\Gamma \vdash [\widehat{\sigma}]N \bullet \vec{v} \Rrightarrow [\widehat{\sigma}]M$ holds by the induction hypothesis as shown above, since $\Xi' \vdash \widehat{\sigma} : C$ implies $\Xi' \vdash \widehat{\sigma} : C_2$, and then $\Xi' \vdash \widehat{\sigma}|_{\mathtt{fav}(N)\cup\mathtt{fav}(M)} : C_2$ and $\Xi \vdash \widehat{\sigma}|_{\mathtt{fav}(N)\cup\mathtt{fav}(M)} : \mathtt{fav}(N) \cup \mathtt{fav}(M)$.

Case 15. $(\forall^{\mathrm{INF}}_{\bullet\Rightarrow})$

By assumption:

1. $\Theta \vdash^{\supseteq} \Xi$,
2. $\Theta\,;\mathtt{dom}\,(\Xi) \vdash \forall\overrightarrow{\alpha^{\mp}}.\, N$ is free from negative algorithmic variables,
3. $\Theta\,;\Gamma\,;\Xi \vDash \forall\overrightarrow{\alpha^{\mp}}.\, N \bullet \vec{v} \Rrightarrow M \dashv \Xi'\,;C$, which by inversion means $\vec{v} \neq$, $\overrightarrow{\alpha^{\mp}} \neq \cdot$, and $\Theta\,;\Gamma\,;\Xi, \overrightarrow{\widehat{\alpha}^{\mp}}\{\Theta\} \vDash [\overrightarrow{\widehat{\alpha}^{\mp}}/\overrightarrow{\alpha^{\mp}}]N \bullet \vec{v} \Rrightarrow M \dashv \Xi'\,;C$. It is easy to see that the induction hypothesis is applicable to the latter judgment:
   - $\Theta \vdash^{\supseteq} \Xi, \overrightarrow{\widehat{\alpha}^{\mp}}\{\Theta\}$ holds by $\Theta \vdash^{\supseteq} \Xi$,
   - $\Theta\,;\mathtt{dom}\,(\Xi), \overrightarrow{\widehat{\alpha}^{\mp}} \vdash [\overrightarrow{\widehat{\alpha}^{\mp}}/\overrightarrow{\alpha^{\mp}}]N$ holds since $\Theta\,;\mathtt{dom}\,(\Xi) \vdash \forall\overrightarrow{\alpha^{\mp}}.\, N$ $[\overrightarrow{\widehat{\alpha}^{\mp}}/\overrightarrow{\alpha^{\mp}}]N$ is normalized and free from negative algorithmic variables since so is N;

This way, by the inductive hypothesis applied to $\Theta\,;\Gamma\,;\Xi,\overrightarrow{\widehat{\alpha}^+}\{\Theta\} \vdash [\overrightarrow{\widehat{\alpha}^+}/\overrightarrow{\alpha^+}]N \bullet \vec{v} \Longrightarrow M \dashv \Xi'\,;C$, we have:

a. $\Theta \vdash^{\supseteq} \Xi'$,

b. $\Xi, \overrightarrow{\widehat{\alpha}^+}\{\Theta\} \subseteq \Xi'$,

c. $\Theta\,;\mathtt{dom}\,(\Xi') \vdash M$ is normalized and free from negative algorithmic variables,

d. $\mathtt{dom}\,(\Xi, \overrightarrow{\widehat{\alpha}^+}\{\Theta\}) \cap \mathtt{fav}(M) \subseteq \mathtt{fav}([\overrightarrow{\widehat{\alpha}^+}/\overrightarrow{\alpha^+}]N)$,

e. $\Xi'|_{\widehat{\Theta}\cup\mathtt{fav}(N)\cup\mathtt{fav}(M)} \vdash C$, where $\widehat{\Theta}$ denotes $\mathtt{fav}([\overrightarrow{\widehat{\alpha}^+}/\overrightarrow{\alpha^+}]N) \cap \overrightarrow{\widehat{\alpha}^+}$, that is the algorithmization of the $\forall$-variables that are actually used in N.

f. for any $\widehat{\sigma}$ such that $\Xi' \vdash \widehat{\sigma} : \widehat{\Theta} \cup \mathtt{fav}(N) \cup \mathtt{fav}(M)$ and $\Xi' \vdash \widehat{\sigma} : C$, we have $\Theta\,;\Gamma \vdash [\widehat{\sigma}][\overrightarrow{\widehat{\alpha}^+}/\overrightarrow{\alpha^+}]N \bullet \vec{v} \Longrightarrow [\widehat{\sigma}]M$.

Let us show the required properties:

1. $\Theta \vdash^{\supseteq} \Xi'$ is shown above;
2. $\Xi \subseteq \Xi'$ since $\Xi, \overrightarrow{\widehat{\alpha}^+}\{\Theta\} \subseteq \Xi'$;
3. $\Theta\,;\mathtt{dom}\,(\Xi') \vdash M$ is normalized and free from negative algorithmic variables, as shown above;
4. $\mathtt{dom}\,(\Xi) \cap \mathtt{fav}(M) \subseteq \mathtt{fav}(N)$ since $\mathtt{dom}\,(\Xi, \overrightarrow{\widehat{\alpha}^+}\{\Theta\}) \cap \mathtt{fav}(M) \subseteq \mathtt{fav}([\overrightarrow{\widehat{\alpha}^+}/\overrightarrow{\alpha^+}]N)$ implies $(\mathtt{dom}\,(\Xi) \cup \overrightarrow{\widehat{\alpha}^+}) \cap \mathtt{fav}(M) \subseteq \mathtt{fav}(N) \cup \overrightarrow{\widehat{\alpha}^+}$, thus, $\mathtt{dom}\,(\Xi) \cap \mathtt{fav}(M) \subseteq \mathtt{fav}(N) \cup \overrightarrow{\widehat{\alpha}^+}$, and since $\mathtt{dom}\,(\Xi)$ is disjoint with $\overrightarrow{\widehat{\alpha}^+}$, $\mathtt{dom}\,(\Xi) \cap \mathtt{fav}(M) \subseteq \mathtt{fav}(N)$;
5. $\Xi'|_{\mathtt{fav}(N)\cup\mathtt{fav}(M)} \vdash C|_{\mathtt{fav}(N)\cup\mathtt{fav}(M)}$ follows from $\Xi'|_{\widehat{\Theta}\cup\mathtt{fav}(N)\cup\mathtt{fav}(M)} \vdash C$ if we restrict both sides to $\mathtt{fav}(N) \cup \mathtt{fav}(M)$.
6. Let us assume $\Xi' \vdash \widehat{\sigma} : \mathtt{fav}(N) \cup \mathtt{fav}(M)$ and $\Xi' \vdash \widehat{\sigma} : C|_{\mathtt{fav}(N)\cup\mathtt{fav}(M)}$. Then to show $\Theta\,;\Gamma \vdash [\widehat{\sigma}]\forall\overrightarrow{\alpha^+}.\, N \bullet \vec{v} \Longrightarrow [\widehat{\sigma}]M$, that is $\Theta\,;\Gamma \vdash \forall\overrightarrow{\alpha^+}.\, [\widehat{\sigma}]N \bullet \vec{v} \Longrightarrow [\widehat{\sigma}]M$, we apply the corresponding declarative rule $(\forall^{\mathrm{INF}}_{\bullet\Longrightarrow})$. To do so, we need to provide a substitution for $\overrightarrow{\alpha^+}$, i.e. $\Theta \vdash \sigma_0 : \overrightarrow{\alpha^+}$ such that $\Theta\,;\Gamma \vdash [\sigma_0][\widehat{\sigma}]N \bullet \vec{v} \Longrightarrow > [\widehat{\sigma}]M$.

   By Lemma 84, we construct $\widehat{\sigma}_0$ such that $\Xi' \vdash \widehat{\sigma}_0 : \overrightarrow{\widehat{\alpha}^+}$ and $\Xi' \vdash \widehat{\sigma}_0 : C|_{\overrightarrow{\widehat{\alpha}^+}}$.

   Then σ_0 is defined as $\widehat{\sigma}_0 \circ \widehat{\sigma}|_{\overrightarrow{\widehat{\alpha}^+}} \circ \overrightarrow{\widehat{\alpha}^+}/\overrightarrow{\alpha^+}$.

   Let us show that the premises of $(\forall^{\mathrm{INF}}_{\bullet\Longrightarrow})$ hold:

   - To show $\Theta \vdash \sigma_0 : \overrightarrow{\alpha^+}$, let us take $\alpha_i^+ \in \overrightarrow{\alpha^+}$. If $\widehat{\alpha}_i^+ \in \mathtt{fav}(M)$ then $[\sigma_0]\alpha_i^+ = [\widehat{\sigma}]\widehat{\alpha}_i^+$, and $\Xi' \vdash \widehat{\sigma} : \mathtt{fav}(N) \cup \mathtt{fav}(M)$ implies $\Xi'(\widehat{\alpha}^+) \vdash [\widehat{\sigma}]\widehat{\alpha}^+$. Analogously, if $\widehat{\alpha}_i^+ \in \overrightarrow{\widehat{\alpha}^+} \setminus \mathtt{fav}(M)$ then $[\sigma_0]\alpha_i^+ = [\widehat{\sigma}_0]\widehat{\alpha}_i^+$, and $\Xi' \vdash \widehat{\sigma}_0 : \overrightarrow{\widehat{\alpha}^+}$ implies $\Xi'(\widehat{\alpha}_i^+) \vdash [\widehat{\sigma}_0]\widehat{\alpha}_i^+$. In any case, $\Xi'(\widehat{\alpha}_i^+) \vdash [\sigma]\alpha_i^+$ can be weakened to $\Theta \vdash [\sigma_0]\alpha_i^+$, since $\Theta \vdash^{\supseteq} \Xi'$.

- Let us show $\Theta\,;\Gamma \vdash [\sigma_0][\widehat{\sigma}]N \bullet \vec{v} \Rrightarrow [\widehat{\sigma}]M$. It suffices to construct $\widehat{\sigma}_1$ such that

  a. $\Xi' \vdash \widehat{\sigma}_1 : \widehat{\Theta} \cup \mathtt{fav}(N) \cup \mathtt{fav}(M)$,

  b. $\Xi' \vdash \widehat{\sigma}_1 : C$,

  c. $[\sigma_0][\widehat{\sigma}]N = [\widehat{\sigma}_1][\overrightarrow{\widehat{\alpha}^+}/\overrightarrow{\alpha^+}]N$, and

  d. $[\widehat{\sigma}]M = [\widehat{\sigma}_1]M$,

  because then we can apply the induction hypothesis (3f) to $\widehat{\sigma}_1$, rewrite the conclusion by $[\widehat{\sigma}_1][\overrightarrow{\widehat{\alpha}^+}/\overrightarrow{\alpha^+}]N = [\sigma_0][\widehat{\sigma}]N$ and $[\widehat{\sigma}_1]M = [\widehat{\sigma}]M$, and infer the required judgement.

  Let us take $\widehat{\sigma}_1 = (\widehat{\sigma}_0 \circ \widehat{\sigma})|_{\widehat{\Theta} \cup \mathtt{fav}(N) \cup \mathtt{fav}(M)}$, then

  a. $\Xi' \vdash \widehat{\sigma}_1 : \widehat{\Theta} \cup \mathtt{fav}(N) \cup \mathtt{fav}(M)$, since $\Xi' \vdash \widehat{\sigma}_0 : \overrightarrow{\widehat{\alpha}^+}$ and $\Xi' \vdash \widehat{\sigma} : \mathtt{fav}(N) \cup \mathtt{fav}(M)$, we have $\Xi' \vdash \widehat{\sigma}_0 \circ \widehat{\sigma} : \overrightarrow{\widehat{\alpha}^+} \cup \mathtt{fav}(N) \cup \mathtt{fav}(M)$, which we restrict to $\widehat{\Theta} \cup \mathtt{fav}(N) \cup \mathtt{fav}(M)$.

  b. $\Xi' \vdash \widehat{\sigma}_1 : C$, Let us take any constraint $e \in C$ restricting variable $\widehat{\beta}^{\pm}$. $\Xi'|_{\widehat{\Theta} \cup \mathtt{fav}(N) \cup \mathtt{fav}(M)} \vdash C$ implies that $\widehat{\beta}^{\pm} \in \widehat{\Theta} \cup \mathtt{fav}(N) \cup \mathtt{fav}(M)$.

  If $\widehat{\beta}^{\pm} \in \mathtt{fav}(N) \cup \mathtt{fav}(M)$ then $[\widehat{\sigma}_1]\widehat{\beta}^{\pm} = [\widehat{\sigma}]\widehat{\beta}^{\pm}$. Additionally, $e \in C|_{\mathtt{fav}(N) \cup \mathtt{fav}(M)}$, which, since $\Xi' \vdash \widehat{\sigma} : C|_{\mathtt{fav}(N) \cup \mathtt{fav}(M)}$, means $\Xi'(\widehat{\beta}^{\pm}) \vdash [\widehat{\sigma}]\widehat{\beta}^{\pm} : e$.

  If $\widehat{\beta}^{\pm} \in \widehat{\Theta} \setminus (\mathtt{fav}(N) \cup \mathtt{fav}(M))$ then $[\widehat{\sigma}_1]\widehat{\beta}^{\pm} = [\widehat{\sigma}_0]\widehat{\beta}^{\pm}$. Additionally, $e \in C|_{\overrightarrow{\widehat{\alpha}^+}}$, which, since $\Xi' \vdash \widehat{\sigma}_0 : C|_{\overrightarrow{\widehat{\alpha}^+}}$, means $\Xi'(\widehat{\beta}^{\pm}) \vdash [\widehat{\sigma}_0]\widehat{\beta}^{\pm} : e$.

  c. Let us prove $[\sigma_0][\widehat{\sigma}]N = [\widehat{\sigma}_1][\overrightarrow{\widehat{\alpha}^+}/\overrightarrow{\alpha^+}]N$ by the following reasoning

$$
\begin{aligned}
[\sigma_0][\widehat{\sigma}]N &= [\widehat{\sigma}_0][\widehat{\sigma}|_{\overrightarrow{\widehat{\alpha}^+}}][\overrightarrow{\widehat{\alpha}^+}/\overrightarrow{\alpha^+}][\widehat{\sigma}]N \\
&\quad \text{by definition of } \sigma_0 \\
&= [\widehat{\sigma}_0][\widehat{\sigma}|_{\overrightarrow{\widehat{\alpha}^+}}][\overrightarrow{\widehat{\alpha}^+}/\overrightarrow{\alpha^+}][\widehat{\sigma}|_{\mathtt{fav}(N)}]N \\
&\quad \text{by Lemma 63} \\
&= [\widehat{\sigma}_0][\widehat{\sigma}|_{\overrightarrow{\widehat{\alpha}^+}}][\widehat{\sigma}|_{\mathtt{fav}(N)}][\overrightarrow{\widehat{\alpha}^+}/\overrightarrow{\alpha^+}]N \\
&\quad \mathtt{fav}(N) \cap \overrightarrow{\widehat{\alpha}^+} = \emptyset \text{ and } \overrightarrow{\alpha^+} \cap \Theta = \emptyset \\
&= [\widehat{\sigma}|_{\overrightarrow{\widehat{\alpha}^+}}][\widehat{\sigma}|_{\mathtt{fav}(N)}][\overrightarrow{\widehat{\alpha}^+}/\overrightarrow{\alpha^+}]N \\
&\quad \text{since } [\widehat{\sigma}|_{\overrightarrow{\widehat{\alpha}^+}}][\widehat{\sigma}|_{\mathtt{fav}(N)}][\overrightarrow{\widehat{\alpha}^+}/\overrightarrow{\alpha^+}]N \text{ is ground} \\
&= [\widehat{\sigma}|_{\overrightarrow{\widehat{\alpha}^+} \cup \mathtt{fav}(N)}][\overrightarrow{\widehat{\alpha}^+}/\overrightarrow{\alpha^+}]N \\
&= [\widehat{\sigma}|_{\widehat{\Theta} \cup \mathtt{fav}(N)}][\overrightarrow{\widehat{\alpha}^+}/\overrightarrow{\alpha^+}]N \\
&\quad \text{by Lemma 63: } \mathtt{fav}([\overrightarrow{\widehat{\alpha}^+}/\overrightarrow{\alpha^+}]N) = \widehat{\Theta} \cup \mathtt{fav}(N) \\
&= [\widehat{\sigma}|_{\widehat{\Theta} \cup \mathtt{fav}(N) \cup \mathtt{fav}(M)}][\overrightarrow{\widehat{\alpha}^+}/\overrightarrow{\alpha^+}]N
\end{aligned}
$$

also by Lemma 63

$$= [(\widehat{\sigma}_0 \circ \widehat{\sigma})|_{\widehat{\Theta} \cup \mathtt{fav}(N) \cup \mathtt{fav}(M)}][\overrightarrow{\widehat{\alpha}^+}/\overrightarrow{\alpha^+}]N$$

$[\widehat{\sigma}|_{\widehat{\Theta} \cup \mathtt{fav}(N) \cup \mathtt{fav}(M)}][\overrightarrow{\widehat{\alpha}^+}/\overrightarrow{\alpha^+}]N$ is ground

$$= [\widehat{\sigma}_1][\overrightarrow{\widehat{\alpha}^+}/\overrightarrow{\alpha^+}]N$$

by definition of $\widehat{\sigma}_1$

d. $[\widehat{\sigma}]M = [\widehat{\sigma}_1]M$ By definition of $\widehat{\sigma}_1$, $[\widehat{\sigma}_1]M$ is equal to $[(\widehat{\sigma}_0 \circ \widehat{\sigma})|_{\widehat{\Theta} \cup \mathtt{fav}(N) \cup \mathtt{fav}(M)}]M$, which by Lemma 63 is equal to $[\widehat{\sigma}_0 \circ \widehat{\sigma}]M$, that is $[\widehat{\sigma}_0][\widehat{\sigma}]M$, and since $[\widehat{\sigma}]M$ is ground, $[\widehat{\sigma}_0][\widehat{\sigma}]M = [\widehat{\sigma}]M$.

- $\overrightarrow{\alpha^+} \neq \cdot$ and $\overrightarrow{v} \neq$ hold by assumption.

■

Lemma 102 (Completeness of Typing)**.** *Suppose that* $\Theta \vdash \Gamma$. *For an inference tree* T_1,

+ *If* T_1 *infers* $\Theta; \Gamma \vdash v : P$ *then* $\Theta; \Gamma \vDash v : \mathtt{nf}\,(P)$
− *If* T_1 *infers* $\Theta; \Gamma \vdash c : N$ *then* $\Theta; \Gamma \vDash c : \mathtt{nf}\,(N)$
• *If* T_1 *infers* $\Theta\,; \Gamma \vdash [\widehat{\sigma}]N \bullet \overrightarrow{v} \Longrightarrow M$ *and*
  1. $\Theta \vdash^{\supseteq} \Xi$,
  2. $\Theta \vdash M$,
  3. $\Theta\,; \mathtt{dom}\,(\Xi) \vdash N$ *(free from negative algorithmic variables, that is* $\widehat{\alpha}^- \notin \mathtt{fav} N$*), and*
  4. $\Xi \vdash \widehat{\sigma} : \mathtt{fav}(N)$,

then there exist M', Ξ', *and* C *such that*

1. $\Theta\,; \Gamma\,; \Xi \vDash N \bullet \overrightarrow{v} \Longrightarrow M' \dashv \Xi'\,; C$ *and*
2. *for any* $\Xi \vdash \widehat{\sigma} : \mathtt{fav}(N)$ *and* $\Theta \vdash M$ *such that* $\Theta\,; \Gamma \vdash [\widehat{\sigma}]N \bullet \overrightarrow{v} \Longrightarrow M$, *there exists* $\widehat{\sigma}'$ *such that*
   a. $\Xi' \vdash \widehat{\sigma}' : \mathtt{fav} N \cup \mathtt{fav} M'$ *and* $\Xi' \vdash \widehat{\sigma}' : C$,
   b. $\Xi \vdash \widehat{\sigma}' \simeq^{\leqslant} \widehat{\sigma} : \mathtt{fav} N$, *and*
   c. $\Theta \vdash [\widehat{\sigma}']M' \simeq^{\leqslant} M$.

Proof We prove it by induction on $\mathsf{metric}(T_1)$, mutually with the soundness of typing (Lemma 101). Let us consider the last rule applied to infer the derivation.

Case 1. $(\{\}^{\mathrm{INF}})$
Then we are proving that if $\Theta; \Gamma \vdash \{c\} : \downarrow N$ (inferred by $(\{\}^{\mathrm{INF}})$) then $\Theta; \Gamma \vDash \{c\} : \mathtt{nf}\,(\downarrow N)$. By inversion of $\Theta; \Gamma \vdash \{c\} : \downarrow N$, we have $\Theta; \Gamma \vdash c : N$, which we apply

the induction hypothesis to to obtain $\Theta; \Gamma \vDash c : \mathtt{nf}\,(N)$. Then by $(\{\}^{\text{INF}})$, we have $\Theta; \Gamma \vDash \{c\} : \downarrow\mathtt{nf}\,(N)$. It is left to notice that $\downarrow\mathtt{nf}\,(N) = \mathtt{nf}\,(\downarrow N)$.

Case 2. $(\text{RET}^{\text{INF}})$

The proof is symmetric to the previous case (case 1).

Case 3. $(\text{ANN}_{+}^{\text{INF}})$

Then we are proving that if $\Theta; \Gamma \vdash (v : Q) : Q$ is inferred by $(\text{ANN}_{+}^{\text{INF}})$ then $\Theta; \Gamma \vDash (v : Q) : \mathtt{nf}\,(Q)$. By inversion, we have:

1. $\Theta \vdash Q$;
2. $\Theta; \Gamma \vdash v : P$, which by the induction hypothesis implies $\Theta; \Gamma \vDash v : \mathtt{nf}\,(P)$;
3. $\Theta \vdash Q \geqslant P$, and by transitivity, $\Theta \vdash Q \geqslant \mathtt{nf}\,(P)$; Since Q is ground, we have $\Theta; \cdot \vdash Q$ and $\Theta \vdash [\cdot]\,Q \geqslant \mathtt{nf}\,(P)$. Then by the completeness of subtyping (Lemma 87), we have $\Theta; \cdot \vDash Q \geqslant \mathtt{nf}\,(P) \dashv C$, where $\cdot \vdash C$ (implying $C = \cdot$). This way, $\Theta; \cdot \vDash Q \geqslant \mathtt{nf}\,(P) \dashv \cdot$.

Then we can apply $(\text{ANN}_{+}^{\text{INF}})$ to $\Theta \vdash Q$, $\Theta; \Gamma \vDash v : \mathtt{nf}\,(P)$ and $\Theta; \cdot \vDash Q \geqslant \mathtt{nf}\,(P) \dashv \cdot$ to infer $\Theta; \Gamma \vDash (v : Q) : \mathtt{nf}\,(Q)$.

Case 4. $(\text{ANN}_{-}^{\text{INF}})$

The proof is symmetric to the previous case (case 3).

Case 5. (λ^{INF})

Then we are proving that if $\Theta; \Gamma \vdash \lambda x : P.\, c : P \to N$ is inferred by (λ^{INF}), then $\Theta; \Gamma \vDash \lambda x : P.\, c : \mathtt{nf}\,(P \to N)$.

By inversion of $\Theta; \Gamma \vdash \lambda x : P.\, c : P \to N$, we have $\Theta \vdash P$ and $\Theta; \Gamma, x : P \vdash c : N$. Then by the induction hypothesis, $\Theta; \Gamma, x : P \vDash c : \mathtt{nf}\,(N)$. By (λ^{INF}), we infer $\Theta; \Gamma \vDash \lambda x : P.\, c : \mathtt{nf}\,(P \to \mathtt{nf}\,(N))$. By idempotence of normalization (Lemma 46), $\mathtt{nf}\,(P \to \mathtt{nf}\,(N)) = \mathtt{nf}\,(P \to N)$, which concludes the proof for this case.

Case 6. (Λ^{INF})

Then we are proving that if $\Theta; \Gamma \vdash \Lambda\alpha^{+}.\, c : \forall\alpha^{+}.\, N$ is inferred by (Λ^{INF}), then $\Theta; \Gamma \vDash \Lambda\alpha^{+}.\, c : \mathtt{nf}\,(\forall\alpha^{+}.\, N)$. Similar to the previous case, by inversion of $\Theta; \Gamma \vdash \Lambda\alpha^{+}.\, c : \forall\alpha^{+}.\, N$, we have $\Theta, \alpha^{+}; \Gamma \vdash c : N$, and then by the induction hypothesis, $\Theta, \alpha^{+}; \Gamma \vDash c : \mathtt{nf}\,(N)$. After that, application of (Λ^{INF}), gives as $\Theta; \Gamma \vDash \Lambda\alpha^{+}.\, c : \mathtt{nf}\,(\forall\alpha^{+}.\, \mathtt{nf}\,(N))$.

It is left to show that $\mathtt{nf}\,(\forall\alpha^{+}.\, \mathtt{nf}\,(N)) = \mathtt{nf}\,(\forall\alpha^{+}.\, N)$. Assume $N = \forall\overrightarrow{\beta^{+}}.\, M$ (where M does not start with $\forall$).

- Then by definition, $\mathtt{nf}\,(\forall\alpha^{+}.\, N) = \mathtt{nf}\,(\forall\alpha^{+}, \overrightarrow{\beta^{+}}.\, M) = \forall\overrightarrow{\gamma^{+}}.\, \mathtt{nf}\,(M)$, where $\mathtt{ord}\,\alpha^{+}, \overrightarrow{\beta^{+}}\ \mathtt{in}\ \mathtt{nf}\,(M) = \overrightarrow{\gamma^{+}}$.
- On the other hand, $\mathtt{nf}\,(N) = \forall\overrightarrow{\gamma^{+\prime}}.\, \mathtt{nf}\,(M)$, where $\mathtt{ord}\,\overrightarrow{\beta^{+}}\ \mathtt{in}\ \mathtt{nf}\,(M) = \overrightarrow{\gamma^{+\prime}}$, and thus, $\mathtt{nf}\,(\forall\alpha^{+}.\, \mathtt{nf}\,(N)) = \mathtt{nf}\,(\forall\alpha^{+}, \overrightarrow{\gamma^{+\prime}}.\, \mathtt{nf}\,(M)) = \forall\overrightarrow{\gamma^{+\prime\prime}}.\, \mathtt{nf}\,(\mathtt{nf}\,(M)) = \forall\overrightarrow{\gamma^{+\prime\prime}}.\, \mathtt{nf}\,(M)$, where $\mathtt{ord}\,\alpha^{+}, \overrightarrow{\gamma^{+\prime}}\ \mathtt{in}\ \mathtt{nf}\,(\mathtt{nf}\,(M)) = \overrightarrow{\gamma^{+\prime\prime}}$.

It is left to show that $\overrightarrow{\gamma^{+}}'' = \overrightarrow{\gamma^{+}}$.

$$
\begin{aligned}
\overrightarrow{\gamma^{+}}'' &= \mathtt{ord}\,\alpha^{+}, \overrightarrow{\gamma^{+}}'\ \mathtt{in}\,\mathtt{nf}\,(\mathtt{nf}\,(M)) \\
&= \mathtt{ord}\,\alpha^{+}, \overrightarrow{\gamma^{+}}'\ \mathtt{in}\,\mathtt{nf}\,(M) \\
&\quad \text{by idempotence (Lemma 46)} \\
&= \mathtt{ord}\,\alpha^{+} \cup \overrightarrow{\beta^{+}} \cap \mathtt{fv}\,\mathtt{nf}\,(M)\ \mathtt{in}\,\mathtt{nf}\,(M) \\
&\quad \text{by definition of } \overrightarrow{\gamma^{+}}' \text{ and Lemma 35} \\
&= \mathtt{ord}\,(\alpha^{+} \cup \overrightarrow{\beta^{+}} \cap \mathtt{fv}\,\mathtt{nf}\,(M)) \cap \mathtt{fv}\,\mathtt{nf}\,(M)\ \mathtt{in}\,\mathtt{nf}\,(M) && \text{by Lemma 36} \\
&= \mathtt{ord}\,(\alpha^{+} \cup \overrightarrow{\beta^{+}}) \cap \mathtt{fv}\,\mathtt{nf}\,(M)\ \mathtt{in}\,\mathtt{nf}\,(M) && \text{by set properties} \\
&= \mathtt{ord}\,\alpha^{+}, \overrightarrow{\beta^{+}}\ \mathtt{in}\,\mathtt{nf}\,(M) \\
&= \overrightarrow{\gamma^{+}}
\end{aligned}
$$

Case 7. ($\textsc{let}^{\textsc{inf}}_{\exists}$)

Then we are proving that if $\Theta; \Gamma \vdash \mathtt{let}^{\exists}(\overrightarrow{\alpha^{-}}, x) = v;\ c : N$ is inferred by ($\textsc{let}^{\textsc{inf}}_{\exists}$), then $\Theta; \Gamma \vDash \mathtt{let}^{\exists}(\overrightarrow{\alpha^{-}}, x) = v;\ c : \mathtt{nf}\,(N)$.

By inversion of $\Theta; \Gamma \vdash \mathtt{let}^{\exists}(\overrightarrow{\alpha^{-}}, x) = v;\ c : N$, we have

1. $\mathtt{nf}\,(\exists \overrightarrow{\alpha^{-}}.\ P) = \exists \overrightarrow{\alpha^{-}}.\ P$,
2. $\Theta; \Gamma \vdash v : \exists \overrightarrow{\alpha^{-}}.\ P$, which by the induction hypothesis implies $\Theta; \Gamma \vDash v : \mathtt{nf}\,(\exists \overrightarrow{\alpha^{-}}.\ P)$, and hence, $\Theta; \Gamma \vDash v : \exists \overrightarrow{\alpha^{-}}.\ P$.
3. $\Theta, \overrightarrow{\alpha^{-}}; \Gamma, x : P \vdash c : N$, and by the induction hypothesis, $\Theta, \overrightarrow{\alpha^{-}}; \Gamma, x : P \vDash c : \mathtt{nf}\,(N)$.
4. $\Theta \vdash N$.

This way, we can apply ($\textsc{let}^{\textsc{inf}}_{\exists}$) to infer $\Theta; \Gamma \vDash \mathtt{let}^{\exists}(\overrightarrow{\alpha^{-}}, x) = v;\ c : \mathtt{nf}\,(N)$.

Case 8. ($\simeq^{\textsc{inf}}_{+}$)

Then we are proving that if $\Theta; \Gamma \vdash v : P'$ is inferred by ($\simeq^{\textsc{inf}}_{+}$), then $\Theta; \Gamma \vDash v : \mathtt{nf}\,(P')$. By inversion, $\Theta; \Gamma \vdash v : P$ and $\Theta \vdash P \simeq^{\leqslant} P'$, and the metric of the tree inferring $\Theta; \Gamma \vdash v : P$ is less than the one inferring $\Theta; \Gamma \vdash v : P'$. Then by the induction hypothesis, $\Theta; \Gamma \vDash v : \mathtt{nf}\,(P)$.

By Lemma 48 $\Theta \vdash P \simeq^{\leqslant} P'$ implies $\mathtt{nf}\,(P) = \mathtt{nf}\,(P')$, and thus, $\Theta; \Gamma \vDash v : \mathtt{nf}\,(P)$ can be rewritten to $\Theta; \Gamma \vDash v : \mathtt{nf}\,(P')$.

Case 9. ($\textsc{Var}^{\textsc{inf}}$)

Then we prove that $\Theta; \Gamma \vdash x : P$ implies $\Theta; \Gamma \vDash x : \mathtt{nf}\,(P)$. By inversion of $\Theta; \Gamma \vdash x : P$, we have $x : P \in \Gamma$. Then ($\textsc{Var}^{\textsc{inf}}$) applies to infer $\Theta; \Gamma \vDash x : \mathtt{nf}\,(P)$.

Case 10. ($\textsc{let}^{\textsc{inf}}$)

Then we prove that $\Theta; \Gamma \vdash \mathtt{let}\,x = v(\overrightarrow{v});\ c : N$ implies $\Theta; \Gamma \vDash \mathtt{let}\,x = v(\overrightarrow{v});\ c : \mathtt{nf}\,(N)$.

By inversion of $\Theta; \Gamma \vdash \mathtt{let}\,x = v(\overrightarrow{v});\ c : N$, we have

1. $\Theta; \Gamma \vdash v : P$, and by the induction hypothesis, $\Theta; \Gamma \vDash v : \mathtt{nf}\,(P)$.
2. $\Theta; \Gamma, x : P \vdash c : N$, and by Lemma 50, since $\Theta \vdash P \simeq^{\leqslant} \mathtt{nf}\,(P)$, we have $\Theta; \Gamma, x : \mathtt{nf}\,(P) \vdash c : N$. Then by the induction hypothesis, $\Theta; \Gamma, x : \mathtt{nf}\,(P) \vDash c : \mathtt{nf}\,(N)$.

Together, $\Theta; \Gamma \vDash v : \mathtt{nf}\,(P)$ and $\Theta; \Gamma, x : \mathtt{nf}\,(P) \vDash c : \mathtt{nf}\,(N)$ imply $\Theta; \Gamma \vDash \mathtt{let}\, x = v(\vec{v});\ c : \mathtt{nf}\,(N)$ by ($\textsc{let}^{\textsc{inf}}$).

Case 11. ($\textsc{let}^{\textsc{inf}}_{\textsc{c}}$)
Then we prove that $\Theta; \Gamma \vdash \mathtt{let}\, x : P = c;\ c' : N$ implies $\Theta; \Gamma \vDash \mathtt{let}\, x : P = c;\ c' : \mathtt{nf}\,(N)$.

By inversion of $\Theta; \Gamma \vdash \mathtt{let}\, x : P = c;\ c' : N$, we have

1. $\Theta \vdash P$;
2. $\Theta; \Gamma \vdash c : M$, and by the induction hypothesis, $\Theta; \Gamma \vDash c : \mathtt{nf}\,(M)$;
3. $\Theta \vdash M \leqslant \uparrow P$, which by Corollary 16 and Lemma 24 implies $\Theta \vdash \mathtt{nf}\,(M) \leqslant \uparrow P$, that is $\Theta \vdash [\cdot]\mathtt{nf}\,(M) \leqslant \uparrow P$. Then by the completeness of subtyping (Lemma 93), we have $\Theta;\ \cdot \vDash \mathtt{nf}\,(M) \leqslant \uparrow P \dashv \cdot$;
4. $\Theta; \Gamma, x : P \vdash c' : N$, and by the induction hypothesis, $\Theta; \Gamma, x : P \vDash c' : \mathtt{nf}\,(N)$.

Together, these premises imply $\Theta; \Gamma \vDash \mathtt{let}\, x : P = c;\ c' : \mathtt{nf}\,(N)$ by ($\textsc{let}^{\textsc{inf}}_{\textsc{c}}$).

Case 12. ($\textsc{let}^{\textsc{inf}}_{:@}$)
Then we prove that $\Theta; \Gamma \vdash \mathtt{let}\, x : P = v(\vec{v});\ c : N$ implies $\Theta; \Gamma \vDash \mathtt{let}\, x : P = v(\vec{v});\ c : \mathtt{nf}\,(N)$.

By inversion of $\Theta; \Gamma \vdash \mathtt{let}\, x : P = v(\vec{v});\ c : N$, we have

1. $\Theta \vdash P$
2. $\Theta; \Gamma \vdash v : \downarrow M$ for some ground M, which by the induction hypothesis means $\Theta; \Gamma \vDash v : \downarrow \mathtt{nf}\,(M)$
3. $\Theta\,; \Gamma \vdash M \bullet \vec{v} \Rrightarrow M'$. By Lemma 57, since $\Theta \vdash M \simeq^{\leqslant} \mathtt{nf}\,(M)$, we have $\Theta\,; \Gamma \vdash [\cdot]\mathtt{nf}\,(M) \bullet \vec{v} \Rrightarrow M'$, which by the induction hypothesis means that there exist normalized M_0, Ξ, and C_1 such that (noting that M is ground):
   a. $\Theta\,; \Gamma\,; \cdot \vDash \mathtt{nf}\,(M) \bullet \vec{v} \Rrightarrow M_0 \dashv \Xi\,; C_1$, where by the soundness, $\Theta\,; \mathtt{dom}\,(\Xi) \vdash M_0$ and $\Xi \vdash C_1$.
   b. for any $\Theta \vdash M''$ such that $\Theta\,; \Gamma \vdash \mathtt{nf}\,(M) \bullet \vec{v} \Rrightarrow M''$ there exists $\widehat{\sigma}$ such that
      i. $\Xi \vdash \widehat{\sigma} : \mathtt{fav} M_0$, $\Xi \vdash \widehat{\sigma} : C_1$, and
      ii. $\Theta \vdash [\widehat{\sigma}] M_0 \simeq^{\leqslant} M''$,

      In particular, there exists $\widehat{\sigma}_0$ such that $\Xi \vdash \widehat{\sigma}_0 : \mathtt{fav} M_0$, $\Xi \vdash \widehat{\sigma}_0 : C_1$, $\Theta \vdash [\widehat{\sigma}_0] M_0 \simeq^{\leqslant} M'$.
4. $\Theta \vdash M' \leqslant \uparrow P$, and by transitivity, since $\Theta \vdash [\widehat{\sigma}_0] M_0 \simeq^{\leqslant} M'$, we have $\Theta \vdash [\widehat{\sigma}_0] M_0 \leqslant \uparrow P$.

Let us apply Lemma 93 to $\Theta \vdash [\widehat{\sigma}_0]M_0 \leqslant \uparrow P$ and obtain $\Xi \vdash C_2$ such that

a. $\Theta;\ \Xi \vDash [\widehat{\sigma}_0]M_0 \leqslant \uparrow P \dashv C_2$ and

b. $\Xi \vdash \widehat{\sigma}_0 : C_2$.

5. $\Theta; \Gamma, x : P \vdash c : N$, and by the induction hypothesis, $\Theta; \Gamma, x : P \vDash c : \mathtt{nf}\,(N)$.

To infer $\Theta; \Gamma \vDash \mathtt{let}\, x : P = v(\vec{v});\ c : \mathtt{nf}\,(N)$, we apply the corresponding algorithmic rule $(\textsc{let}^{\textsc{inf}}_{:@})$. Let us show that the premises hold:

1. $\Theta \vdash P$,
2. $\Theta; \Gamma \vDash v : \downarrow \mathtt{nf}\,(M)$,
3. $\Theta\,;\Gamma\,;\cdot \vDash \mathtt{nf}\,(M) \bullet \vec{v} \Rrightarrow M_0 \dashv \Xi\,; C_1$,
4. $\Theta;\ \Xi \vDash M_0 \leqslant \uparrow P \dashv C_2$, and
5. $\Theta; \Gamma, x : P \vDash c : \mathtt{nf}\,(N)$ hold as noted above;
6. $\Xi \vdash C_1 \,\&\, C_2 = C$ is defined by Lemma 91, since $\Xi \vdash \widehat{\sigma}_0 : C_1$ and $\Xi \vdash \widehat{\sigma}_0 : C_2$.

Case 13. $(\textsc{let}^{\textsc{inf}}_{@})$

By assumption, c is $\mathtt{let}\, x = v(\vec{v});\ c'$. Then by inversion of $\Theta; \Gamma \vdash \mathtt{let}\, x = v(\vec{v});\ c' : N$:

- $\Theta; \Gamma \vdash v : \downarrow M$, which by the induction hypothesis means $\Theta; \Gamma \vDash v : \downarrow \mathtt{nf}\,(M)$;
- $\Theta\,;\Gamma \vdash M \bullet \vec{v} \Rrightarrow \uparrow Q$ principal. Then by Lemma 57, since $\Theta \vdash M \simeq^{\leqslant} \mathtt{nf}\,(M)$, we have $\Theta\,;\Gamma \vdash \mathtt{nf}\,(M) \bullet \vec{v} \Rrightarrow \uparrow Q$ and moreover, $\Theta\,;\Gamma \vdash \mathtt{nf}\,(M) \bullet \vec{v} \Rrightarrow \uparrow Q$ principal: since for any inference, $\mathtt{nf}\,(M)$ can be replaced back with M, the sets of types Q' inferred for the applications $\Theta\,;\Gamma \vdash \mathtt{nf}\,(M) \bullet \vec{v} \Rrightarrow \uparrow Q'$ and $\Theta\,;\Gamma \vdash M \bullet \vec{v} \Rrightarrow \uparrow Q'$ are the same. Then the induction hypothesis applied to $\Theta\,;\Gamma \vdash [\cdot]\mathtt{nf}\,(M) \bullet \vec{v} \Rrightarrow \uparrow Q$ implies that there exist M', Ξ, and C such that (considering M is ground):

  1. $\Theta\,;\Gamma\,;\cdot \vDash \mathtt{nf}\,(M) \bullet \vec{v} \Rrightarrow M' \dashv \Xi\,; C$, which, by the soundness, implies, in particular that

     a. $\Theta\,;\mathtt{dom}\,(\Xi) \vdash M'$ is normalized and free of negative algorithmic variables,

     b. $\Xi|_{\mathtt{fav}(M')} \vdash C$, which means $\mathtt{dom}\,(C) \subseteq \mathtt{fav}(M')$,

     c. for any $\Xi \vdash \widehat{\sigma} : \mathtt{fav} M'$ such that $\Xi \vdash \widehat{\sigma} : C$, we have $\Theta\,;\Gamma \vdash \mathtt{nf}\,(M) \bullet \vec{v} \Rrightarrow [\widehat{\sigma}]M'$.

     and

  2. for any $\Theta \vdash M''$ such that $\Theta\,;\Gamma \vdash \mathtt{nf}\,(M) \bullet \vec{v} \Rrightarrow M''$, (and in particular, for $\Theta \vdash \uparrow Q$) there exists $\widehat{\sigma}_1$ such that

     a. $\Xi \vdash \widehat{\sigma}_1 : \mathtt{fav} M'$, $\Xi \vdash \widehat{\sigma}_1 : C$, and

b. $\Theta \vdash [\widehat{\sigma}_1]M' \simeq^{\leqslant} M''$, and in particular, $\Theta \vdash [\widehat{\sigma}_1]M' \simeq^{\leqslant} \uparrow Q$. Since M' is normalized and free of negative algorithmic variables, it means that $M' = \uparrow P$ for some P $(\Theta\,;\mathtt{dom}\,(\Xi) \vdash P)$ that is $\Theta \vdash [\widehat{\sigma}_1]P \simeq^{\leqslant} Q$.

- $\Theta; \Gamma, x : Q \vdash c' : N$

To infer $\Theta; \Gamma \vDash \mathtt{let}\, x = v(\vec{v});\ c' : \mathtt{nf}\,(N)$, let us apply the corresponding algorithmic rule ($(\mathrm{LET}^{\mathrm{INF}}_{@})$):

1. $\Theta; \Gamma \vDash v : \downarrow\mathtt{nf}\,(M)$ holds as noted above;
2. $\Theta\,; \Gamma\,; \cdot \vDash \mathtt{nf}\,(M) \bullet \vec{v} \Rrightarrow \uparrow P \dashv \Xi\,; C$ holds as noted above;
3. Let us show that $nf(iQ)$ is the minimal instantiation of P w.r.t. C, in other words, P is C-minimized by $\widehat{\sigma}$ for some $\widehat{\sigma}$ and $[\widehat{\sigma}]P = \mathtt{nf}\,(Q)$. By rewriting $\mathtt{nf}\,(Q)$ as $\mathtt{nf}\,([\widehat{\sigma}_1]P)$, we need to show $[\widehat{\sigma}]P = \mathtt{nf}\,([\widehat{\sigma}_1]P)$.

   Let us apply the completeness of minimal instantiation (Lemma 95). That would give us $\widehat{\sigma} = \mathtt{nf}\,(\widehat{\sigma}_1)$, which would immediately imply the required equality. To do that, we need to demonstrate that $\widehat{\sigma}_1$ is the minimal instantiation of P w.r.t. C. In other words, any other substitution respecting C, instantiate P into a *supertype* of Q. To do that, we apply the principality of Q: $\Theta\,; \Gamma \vdash \mathtt{nf}\,(M) \bullet \vec{v} \Rrightarrow \uparrow Q$ principal: which means that for any other Q' such that $\Theta\,; \Gamma \vdash \mathtt{nf}\,(M) \bullet \vec{v} \Rrightarrow \uparrow Q'$, we have $\Theta \vdash Q' \geqslant Q$. It is left to show that any substitution respecting C gives us Q' inferrable for the application $\Theta\,; \Gamma \vdash M \bullet \vec{v} \Rrightarrow \uparrow Q'$, which holds by 1c.
4. To show $\mathtt{fav} P = \mathtt{dom}\,(C)$ and $C\ \mathtt{singular\ with}\ \widehat{\sigma}_0$ for some $\widehat{\sigma}_0$, we apply Lemma 99 with $\widehat{\Theta} = \mathtt{fav} P = \mathtt{fav}(M')$ (as noted above, $\mathtt{dom}\,(C) \subseteq \mathtt{fav}(M') = \widehat{\Theta}$).

   Now we will show that any substitution satisfying C is equivalent to $\widehat{\sigma}_1$. As noted in 1c, for any substitution $\Xi \vdash \widehat{\sigma} : \widehat{\Theta}$, $\Xi \vdash \widehat{\sigma} : C$ implies $\Theta \vdash [\widehat{\sigma}]M' \simeq^{\leqslant} \uparrow Q$, which is rewritten as $\Theta \vdash [\widehat{\sigma}]P \simeq^{\leqslant} Q$. And since $\Theta \vdash [\widehat{\sigma}_1]P \simeq^{\leqslant} Q$, we have $\Theta \vdash [\widehat{\sigma}]P \simeq^{\leqslant} [\widehat{\sigma}_1]P$, which implies $\Xi \vdash \widehat{\sigma} \simeq^{\leqslant} \widehat{\sigma}_1 : \widehat{\Theta}$ by Corollary 23.
5. Let us show $\Theta; \Gamma, x : [\widehat{\sigma}_0]P \vDash c' : \mathtt{nf}\,(N)$. By the soundness of singularity (Lemma 98), we have $\Xi \vdash \widehat{\sigma}_0 : C$, which by 1c means $\Theta \vdash [\widehat{\sigma}_0]M' \simeq^{\leqslant} \uparrow Q$, that is $\Theta \vdash [\widehat{\sigma}_0]P \simeq^{\leqslant} Q$, and thus, $\Theta \vdash \Gamma, x : Q \simeq^{\leqslant} \Gamma, x : [\widehat{\sigma}_0]P$.

   Then by Lemma 50, $\Theta; \Gamma, x : Q \vdash c' : N$ can be rewritten as $\Theta; \Gamma, x : [\widehat{\sigma}_0]P \vdash c' : N$. Then by the induction hypothesis applied to it, $\Theta; \Gamma, x : [\widehat{\sigma}_0]P \vDash c' : \mathtt{nf}\,(N)$ holds.

Case 14. $(\forall^{\mathrm{INF}}_{\bullet\Rightarrow})$

Since N cannot be a algorithmic variable, if $[\widehat{\sigma}]N$ starts with $\forall$, so does N. This way, $N = \forall\overrightarrow{\alpha^{+}}.\, N_1$. Then by assumption:

1. $\Theta \vdash^{\supseteq} \Xi$
2. $\Theta\,; \mathtt{dom}\,(\Xi) \vdash \forall\overrightarrow{\alpha^{+}}.\, N_1$ is free from negative algorithmic variables, and then $\Theta, \overrightarrow{\alpha^{+}}\,; \mathtt{dom}\,(\Xi) \vdash N_1$ is free from negative algorithmic variables too;

3. $\Xi \vdash \widehat{\sigma} : \mathtt{fav} N_1$;
4. $\Theta \vdash M$;
5. $\Theta ; \Gamma \vdash [\widehat{\sigma}] \forall \overrightarrow{\alpha^+}. N_1 \bullet \overrightarrow{v} \Longrightarrow M$, that is $\Theta ; \Gamma \vdash (\forall \overrightarrow{\alpha^+}. [\widehat{\sigma}] N_1) \bullet \overrightarrow{v} \Longrightarrow M$. Then by inversion there exists σ such that
   a. $\Theta \vdash \sigma : \overrightarrow{\alpha^+}$;
   b. $\overrightarrow{v} \neq$ and $\overrightarrow{\alpha^+} \neq \cdot$; and
   c. $\Theta ; \Gamma \vdash [\sigma][\widehat{\sigma}] N_1 \bullet \overrightarrow{v} \Longrightarrow M$. Notice that σ and $\widehat{\sigma}$ commute because the codomain of σ does not contain algorithmic variables (and thus, does not intersect with the domain of $\widehat{\sigma}$), and the codomain of $\widehat{\sigma}$ is Θ and does not intersect with $\overrightarrow{\alpha^+}$—the domain of σ.

      Let us take fresh $\overrightarrow{\widehat{\alpha}^+}$ and construct $N_0 = [\overrightarrow{\widehat{\alpha}^+}/\overrightarrow{\alpha^+}] N_1$ and $\Xi, \overrightarrow{\widehat{\alpha}^+}\{\Theta\} \vdash \widehat{\sigma}_0 : \mathtt{fav}(N_0)$ defined as

$$\begin{cases} [\widehat{\sigma}_0]\widehat{\alpha}_i^{\,+} = [\sigma]\alpha_i^+ & \text{for } \widehat{\alpha}_i^{\,+} \in \overrightarrow{\widehat{\alpha}^+} \cap \mathtt{fav} N_0 \\ [\widehat{\sigma}_0]\widehat{\beta}^{\pm} = [\widehat{\sigma}]\widehat{\beta}^{\pm} & \text{for } \widehat{\beta}^{\pm} \in \mathtt{fav} N_1 \end{cases}$$

      Then it is easy to see that $[\widehat{\sigma}_0][\overrightarrow{\widehat{\alpha}^+}/\overrightarrow{\alpha^+}] N_1 = [\sigma][\widehat{\sigma}] N_1$ because this substitution compositions coincide on $\mathtt{fav}(N_1) \cup \mathtt{fv}(N_1)$. In other words, $[\widehat{\sigma}_0] N_0 = [\sigma][\widehat{\sigma}] N_1$.

      Then let us apply the induction hypothesis to $\Theta ; \Gamma \vdash [\widehat{\sigma}_0] N_0 \bullet \overrightarrow{v} \Longrightarrow M$ and obtain M', Ξ', and C such that

      - $\Theta ; \Gamma ; \Xi, \overrightarrow{\widehat{\alpha}^+}\{\Theta\} \vdash N_0 \bullet \overrightarrow{v} \Longrightarrow M' \dashv \Xi' ; C$ and
      - for any $\Xi, \overrightarrow{\widehat{\alpha}^+}\{\Theta\} \vdash \widehat{\sigma}_0 : \mathtt{fav}(N_0)$ and $\Theta \vdash M$ such that $\Theta ; \Gamma \vdash [\widehat{\sigma}_0] N_0 \bullet \overrightarrow{v} \Longrightarrow M$, there exists $\widehat{\sigma}_0'$ such that
        i. $\Xi' \vdash \widehat{\sigma}_0' : \mathtt{fav}(N_0) \cup \mathtt{fav}(M')$, $\Xi' \vdash \widehat{\sigma}_0' : C$,
        ii. $\Xi, \overrightarrow{\widehat{\alpha}^+}\{\Theta\} \vdash \widehat{\sigma}_0' \simeq^{\leqslant} \widehat{\sigma}_0 : \mathtt{fav} N_0$, and
        iii. $\Theta \vdash [\widehat{\sigma}_0'] M' \simeq^{\leqslant} M$.

Let us take M', Ξ', and C from the induction hypothesis (5c) (from C we subtract entries restricting $\overrightarrow{\widehat{\alpha}^+}$) and show they satisfy the required properties

1. To infer $\Theta ; \Gamma ; \Xi \vdash \forall \overrightarrow{\alpha^+}. N_1 \bullet \overrightarrow{v} \Longrightarrow M' \dashv \Xi' ; C \setminus \overrightarrow{\widehat{\alpha}^+}$ we apply the corresponding algorithmic rule ($\forall^{\mathrm{INF}}_{\bullet\Rightarrow}$). As noted above, the required premises hold:
   a. $\overrightarrow{v} \neq$, $\overrightarrow{\alpha^+} \neq \cdot$; and
   b. $\Theta ; \Gamma ; \Xi, \overrightarrow{\widehat{\alpha}^+}\{\Theta\} \vdash [\overrightarrow{\widehat{\alpha}^+}/\overrightarrow{\alpha^+}] N_1 \bullet \overrightarrow{v} \Longrightarrow M' \dashv \Xi' ; C$ is obtained by unfolding the definition of N_0 in $\Theta ; \Gamma ; \Xi, \overrightarrow{\widehat{\alpha}^+}\{\Theta\} \vdash N_0 \bullet \overrightarrow{v} \Longrightarrow M' \dashv \Xi' ; C$ (5c).

2. Let us take and arbitrary $\Xi \vdash \widehat{\sigma} : \mathtt{fav} N_1$ and $\Theta \vdash M$ and assume $\Theta ; \Gamma \vdash [\widehat{\sigma}]\forall \overrightarrow{\alpha^+}. N_1 \bullet \vec{v} \Rrightarrow M$. Then the same reasoning as in 5c applies. In particular, we construct $\Xi, \overrightarrow{\widehat{\alpha}^+}\{\Theta\} \vdash \widehat{\sigma}_0 : \mathtt{fav}(N_0)$ as an extension of $\widehat{\sigma}$ and obtain $\Theta ; \Gamma \vdash [\widehat{\sigma}_0] N_0 \bullet \vec{v} \Rrightarrow M$.

   It means we can apply the property inferred from the induction hypothesis (5c) to obtain $\widehat{\sigma}'_0$ such that

   a. $\Xi' \vdash \widehat{\sigma}'_0 : \mathtt{fav}(N_0) \cup \mathtt{fav}(M')$ and $\Xi' \vdash \widehat{\sigma}'_0 : C$,

   b. $\Xi, \overrightarrow{\widehat{\alpha}^+}\{\Theta\} \vdash \widehat{\sigma}'_0 \simeq^{\leqslant} \widehat{\sigma}_0 : \mathtt{fav} N_0$, and

   c. $\Theta \vdash [\widehat{\sigma}'_0] M' \simeq^{\leqslant} M$.

   Let us show that $\widehat{\sigma}'_0|_{(\mathtt{fav}(N_1) \cup \mathtt{fav}(M'))}$ satisfies the required properties.

   a. $\Xi' \vdash \widehat{\sigma}'_0|_{(\mathtt{fav}(N_1) \cup \mathtt{fav}(M'))} : (\mathtt{fav}(N_1) \cup \mathtt{fav}(M'))$ holds since $\Xi' \vdash \widehat{\sigma}'_0 : \mathtt{fav}(N_0) \cup \mathtt{fav}(M')$ and $\mathtt{fav}(N_1) \cup \mathtt{fav}(M') \subseteq \mathtt{fav}(N_0) \cup \mathtt{fav}(M')$; $\Xi' \vdash \widehat{\sigma}'_0|_{(\mathtt{fav}(N_1) \cup \mathtt{fav}(M'))} : C \setminus \overrightarrow{\widehat{\alpha}^+}$ holds since $\Xi' \vdash \widehat{\sigma}'_0 : C$, $\Xi' \vdash \widehat{\sigma}'_0 : \mathtt{fav}(N_0) \cup \mathtt{fav}(M')$, and $(\mathtt{fav}(N_0) \cup \mathtt{fav}(M')) \setminus \overrightarrow{\widehat{\alpha}^+} = \mathtt{fav}(N_1) \cup \mathtt{fav}(M')$.

   b. $\Theta \vdash [\widehat{\sigma}'_0] M' \simeq^{\leqslant} M$ holds as shown, and hence it holds for $\widehat{\sigma}'_0|_{(\mathtt{fav}(N_1) \cup \mathtt{fav}(M'))}$;

   c. We show $\Xi \vdash \widehat{\sigma}'_0 \simeq^{\leqslant} \widehat{\sigma} : \mathtt{fav} N_1$, from which it follows that it holds for $\widehat{\sigma}'_0|_{(\mathtt{fav}(N_1) \cup \mathtt{fav}(M'))}$. Let us take an arbitrary $\widehat{\beta}^{\pm} \in \mathtt{dom}(\Xi) \subseteq \mathtt{dom}(\Xi) \cup \overrightarrow{\widehat{\alpha}^+}$. Then since $\Xi, \overrightarrow{\widehat{\alpha}^+}\{\Theta\} \vdash \widehat{\sigma}'_0 \simeq^{\leqslant} \widehat{\sigma}_0 : \mathtt{fav} N_0$, we have $\Xi(\widehat{\beta}^{\pm}) \vdash [\widehat{\sigma}'_0]\widehat{\beta}^{\pm} \simeq^{\leqslant} [\widehat{\sigma}_0]\widehat{\beta}^{\pm}$ and by definition of $\widehat{\sigma}_0$, $[\widehat{\sigma}_0]\widehat{\beta}^{\pm} = [\widehat{\sigma}]\widehat{\beta}^{\pm}$.

Case 15. $(\to^{\mathrm{INF}}_{\bullet\Rightarrow})$

Since N cannot be a algorithmic variable, if the shape of $[\widehat{\sigma}]N$ is an arrow, so is the shape of N. This way, $N = Q \to N_1$. Then by assumption:

1. $\Theta \vdash^{\supseteq} \Xi$;
2. $\Theta ; \mathtt{dom}(\Xi) \vdash Q \to N_1$ is free from negative algorithmic variables;
3. $\Xi \vdash \widehat{\sigma} : \mathtt{fav} Q \cup \mathtt{fav} N_1$;
4. $\Theta \vdash M$;
5. $\Theta ; \Gamma \vdash [\widehat{\sigma}](Q \to N_1) \bullet v, \vec{v} \Rrightarrow M$, that is $\Theta ; \Gamma \vdash ([\widehat{\sigma}] Q \to [\widehat{\sigma}] N_1) \bullet v, \vec{v} \Rrightarrow M$, and by inversion:

   a. $\Theta ; \Gamma \vdash v : P$, and by the induction hypothesis, $\Theta ; \Gamma \vDash v : \mathtt{nf}(P)$;

   b. $\Theta \vdash [\widehat{\sigma}] Q \geqslant P$, which by transitivity (Lemma 24) means $\Theta \vdash [\widehat{\sigma}] Q \geqslant \mathtt{nf}(P)$, and then by completeness of subtyping (Lemma 87), $\Theta ; \Xi \vDash Q \geqslant \mathtt{nf}(P) \dashv C_1$, for some $\Xi \vdash C_1 : \mathtt{fav}(Q)$, and moreover, $\Xi \vdash \widehat{\sigma} : C_1$;

c. $\Theta ; \Gamma \vdash [\widehat{\sigma}]N_1 \bullet \vec{v} \Rrightarrow M$. Notice that the induction hypothesis applies to this case: $\Theta ; \mathtt{dom}\,(\Xi) \vdash N_1$ is free from negative algorithmic variables because so is $Q \to N_1$. This way, there exist M', Ξ', and C_2 such that

i. $\Theta ; \Gamma ; \Xi \vDash N_1 \bullet \vec{v} \Rrightarrow M' \dashv \Xi' ; C_2$ and then by the soundness of typing (i.e. the induction hypothesis),

A. $\Xi \subseteq \Xi'$

B. $\Theta ; \mathtt{dom}\,(\Xi') \vdash M'$

C. $\mathtt{dom}\,(\Xi) \cap \mathtt{fav}(M') \subseteq \mathtt{fav} N_1$

D. $\Xi'|_{\mathtt{fav} N_1 \cup \mathtt{fav} M'} \vdash C_2$

ii. for any $\Xi \vdash \widehat{\sigma} : \mathtt{fav}(N_1)$ and $\Theta \vdash M$ such that $\Theta ; \Gamma \vdash [\widehat{\sigma}]N_1 \bullet \vec{v} \Rrightarrow M$, there exists $\widehat{\sigma}'$ such that

A. $\Xi' \vdash \widehat{\sigma}' : \mathtt{fav}(N_1) \cup \mathtt{fav}(M')$ and $\Xi' \vdash \widehat{\sigma}' : C_2$,

B. $\Xi \vdash \widehat{\sigma}' \simeq^{\leqslant} \widehat{\sigma} : \mathtt{fav}(N_1)$, and

C. $\Theta \vdash [\widehat{\sigma}']M' \simeq^{\leqslant} M$.

We need to show that there exist M', Ξ', and C such that $\Theta ; \Gamma ; \Xi \vDash Q \to N_1 \bullet v, \vec{v} \Rrightarrow > M' \dashv \Xi' ; C$ and the initiality property holds. We take M' and Ξ' from the induction hypothesis (5c), and C as a merge of C_1 and C_2. To show that $\Xi' \vdash C_1 \,\&\, C_2 = C$ exists, we apply Lemma 91. To do so, we need to provide a substitution satisfying both C_1 and C_2.

Notice that $\mathtt{dom}\,(C_1) = \mathtt{fav}(Q)$ and $\mathtt{dom}\,(C_2) \subseteq \mathtt{fav} N_1 \cup \mathtt{fav} M'$. This way, it suffices to construct $\Xi' \vdash \widehat{\sigma}'' : \mathtt{fav}(Q) \cup \mathtt{fav} N_1 \cup \mathtt{fav} M'$ such that $\Xi' \vdash \widehat{\sigma}'' : C_1$ and $\Xi' \vdash \widehat{\sigma}'' : C_2$.

By the induction hypothesis (5(c)ii), $\widehat{\sigma}|_{\mathtt{fav}(N_1)}$ can be extended to $\widehat{\sigma}'$ such that

1. $\Xi' \vdash \widehat{\sigma}' : \mathtt{fav}(N_1) \cup \mathtt{fav}(M')$ and $\Xi' \vdash \widehat{\sigma}' : C_2$,
2. $\Xi \vdash \widehat{\sigma}' \simeq^{\leqslant} \widehat{\sigma} : \mathtt{fav}(N_1)$, and
3. $\Theta \vdash [\widehat{\sigma}']M' \simeq^{\leqslant} M$.

Let us extend $\widehat{\sigma}'$ to $\widehat{\sigma}''$ defined on $\mathtt{fav}(Q) \cup \mathtt{fav}(N_1) \cup \mathtt{fav}(M')$ with values of $\widehat{\sigma}$ as follows:

$$\begin{cases} [\widehat{\sigma}'']\widehat{\beta}^{\pm} = [\widehat{\sigma}']\widehat{\beta}^{\pm} & \text{for } \widehat{\beta}^{\pm} \in \mathtt{fav}(N_1) \cup \mathtt{fav}(M') \\ [\widehat{\sigma}'']\widehat{\gamma}^{\pm} = [\widehat{\sigma}]\widehat{\gamma}^{\pm} & \text{for } \widehat{\gamma}^{\pm} \in \mathtt{fav}(Q) \setminus (\mathtt{fav}(N_1) \cup \mathtt{fav}(M')) \end{cases}$$

First, notice that $\Xi' \vdash \widehat{\sigma}'' \simeq^{\leqslant} \widehat{\sigma}' : \mathtt{fav}(N_1) \cup \mathtt{fav}(M')$ by definition. Then since $\Xi' \vdash \widehat{\sigma}' : C_2$ and $\Xi' \vdash C_2 : \mathtt{fav}(N_1) \cup \mathtt{fav}(M')$, we have $\Xi' \vdash \widehat{\sigma}'' : C_2$.

Second, notice that $\Xi \vdash \widehat{\sigma}'' \simeq^{\leqslant} \widehat{\sigma} : \mathtt{fav}(N_1) \cup \mathtt{fav}(Q)$:

- if $\widehat{\gamma}^{\pm} \in \mathtt{fav}(Q) \setminus (\mathtt{fav}(N_1) \cup \mathtt{fav}(M'))$ then $[\widehat{\sigma}'']\widehat{\gamma}^{\pm} = [\widehat{\sigma}]\widehat{\gamma}^{\pm}$ by definition of $\widehat{\sigma}''$;

- if $\widehat{\gamma}^{\pm} \in \mathtt{fav}(Q) \cap \mathtt{fav}(N_1)$ then $[\widehat{\sigma}'']\widehat{\gamma}^{\pm} = [\widehat{\sigma}']\widehat{\gamma}^{\pm}$, and $\Xi \vdash \widehat{\sigma}' \simeq^{\leqslant} \widehat{\sigma} : \mathtt{fav}(N_1)$, as noted above;
- if $\widehat{\gamma}^{\pm} \in \mathtt{fav}(Q) \cap \mathtt{fav}(M')$ then since $\Theta ; \mathtt{dom}(\Xi) \vdash Q$, we have $\mathtt{fav}(Q) \subseteq \mathtt{dom}(\Xi)$, implying $\widehat{\gamma}^{\pm} \in \mathtt{dom}(\Xi) \cap \mathtt{fav}(M') \subseteq \mathtt{fav}(N_1)$. This way, $\widehat{\gamma}^{\pm} \in \mathtt{fav}(Q) \cap \mathtt{fav}(N_1)$, and this case is covered by the previous one.

In particular, $\Xi \vdash \widehat{\sigma}'' \simeq^{\leqslant} \widehat{\sigma} : \mathtt{fav}(Q)$. Then since $\Xi \vdash \widehat{\sigma} : C_1$ and $\Xi \vdash C_1 : \mathtt{fav}(Q)$, we have $\Xi \vdash \widehat{\sigma}'' : C_1$.

This way, $\widehat{\sigma}'$ satisfies both C_1 and C_2, and by the completeness of constraint merge (Lemma 91), $\Xi' \vdash C_1 \,\&\, C_2 = C$ exists.

Finally, to show the required properties, we take M' and Ξ' from the induction hypothesis (5(c)ii), and C defined above. Then

1. $\Theta ; \Gamma ; \Xi \vDash Q \to N_1 \bullet v, \vec{v} \Rrightarrow M' \dashv \Xi' ; C$ is inferred by $(\to \overset{\text{INF}}{\bullet\Rightarrow})$. As noted above:
   a. $\Theta ; \Gamma \vDash v : \mathtt{nf}(P)$,
   b. $\Theta ; \Xi \vDash Q \geqslant \mathtt{nf}(P) \dashv C_1$,
   c. $\Theta ; \Gamma ; \Xi \vDash N_1 \bullet \vec{v} \Rrightarrow M' \dashv \Xi' ; C_2$, and
   d. $\Xi' \vdash C_1 \,\&\, C_2 = C$.
2. let us take an arbitrary $\Xi \vdash \widehat{\sigma}_0 : \mathtt{fav} Q \cup \mathtt{fav} N_1$; and $\Theta \vdash M_0$; such that $\Theta ; \Gamma \vdash [\widehat{\sigma}_0](Q \to N_1) \bullet v, \vec{v} \Rrightarrow M_0$. Then by inversion of $\Theta ; \Gamma \vdash [\widehat{\sigma}_0] Q \to [\widehat{\sigma}_0] N_1 \bullet v, \vec{v} \Rrightarrow M_0$, we have the same properties as in 5. In particular,
   - $\Theta \vdash [\widehat{\sigma}_0] Q \geqslant \mathtt{nf}(P)$ and by the completeness of subtyping (Lemma 87), $\Xi \vdash \widehat{\sigma}_0 : C_1$.
   - $\Theta ; \Gamma \vdash [\widehat{\sigma}_0] N_1 \bullet \vec{v} \Rrightarrow M_0$. Then by 5(c)ii, there exists $\widehat{\sigma}_0'$ such that
     a. $\Xi' \vdash \widehat{\sigma}_0' : \mathtt{fav}(N_1) \cup \mathtt{fav}(M')$ and $\Xi' \vdash \widehat{\sigma}_0' : C_2$,
     b. $\Xi \vdash \widehat{\sigma}_0' \simeq^{\leqslant} \widehat{\sigma}_0 : \mathtt{fav}(N_1)$, and
     c. $\Theta \vdash [\widehat{\sigma}_0'] M' \simeq^{\leqslant} M_0$.

   Let us extend $\widehat{\sigma}_0'$ to be defined on $\mathtt{fav}(Q) \cup \mathtt{fav}(N_1) \cup \mathtt{fav}(M')$ with the values of $\widehat{\sigma}_0$. We define $\widehat{\sigma}_0''$ as follows:

$$\begin{cases} [\widehat{\sigma}_0'']\widehat{\gamma}^{\pm} = [\widehat{\sigma}_0']\widehat{\gamma}^{\pm} & \text{for } \widehat{\gamma}^{\pm} \in \mathtt{fav}(N_1) \cup \mathtt{fav}(M') \\ [\widehat{\sigma}_0'']\widehat{\gamma}^{\pm} = [\widehat{\sigma}_0]\widehat{\gamma}^{\pm} & \text{for } \widehat{\gamma}^{\pm} \in \mathtt{fav}(Q) \setminus (\mathtt{fav}(N_1) \cup \mathtt{fav}(M')) \end{cases}$$

   This way,
   - $\Xi' \vdash \widehat{\sigma}_0'' : \mathtt{fav}(Q) \cup \mathtt{fav}(N_1) \cup \mathtt{fav}(M')$,
   - $\Xi' \vdash \widehat{\sigma}_0'' : C$, since $\Xi' \vdash \widehat{\sigma}_0'' : C_1$ and $\Xi' \vdash \widehat{\sigma}_0'' : C_2$, which is proved similarly to $\Xi' \vdash \widehat{\sigma}'' : C_1$ and $\Xi' \vdash \widehat{\sigma}'' : C_2$ above;

- $\Xi \vdash \widehat{\sigma}_0'' \simeq^{\leqslant} \widehat{\sigma}_0 : \mathtt{fav}(N_1) \cup \mathtt{fav}(Q)$: the proof is analogous to $\Xi \vdash \widehat{\sigma}'' \simeq^{\leqslant} \widehat{\sigma} : \mathtt{fav}(N_1) \cup \mathtt{fav}(Q)$ above.
- $\Theta \vdash [\widehat{\sigma}_0'']M' \simeq^{\leqslant} M_0$ Notice that $\Xi' \vdash \widehat{\sigma}_0'' \simeq^{\leqslant} \widehat{\sigma}_0' : \mathtt{fav}(N_1) \cup \mathtt{fav}(M')$, which is proved analogously to $\Xi' \vdash \widehat{\sigma}'' \simeq^{\leqslant} \widehat{\sigma}' : \mathtt{fav}(N_1) \cup \mathtt{fav}(M')$ above. Then $\Theta \vdash [\widehat{\sigma}_0']M' \simeq^{\leqslant} M_0$ can be rewritten to $\Theta \vdash [\widehat{\sigma}_0'']M' \simeq^{\leqslant} M_0$.

Case 16. ($\emptyset^{\mathrm{INF}}_{\bullet\Rightarrow}$)

By assumption:

1. $\Theta \vdash^{\supseteq} \Xi$,
2. $\Theta \vdash N'$,
3. $\Theta\,;\mathtt{dom}\,(\Xi) \vdash N$ and N is free from negative variables,
4. $\Xi \vdash \widehat{\sigma} : \mathtt{fav}(N)$,
5. $\Theta\,;\Gamma \vdash [\widehat{\sigma}]N \bullet \Rightarrow N'$, and by inversion, $\Theta \vdash [\widehat{\sigma}]N \simeq^{\leqslant} N'$.

Then we can apply the corresponding algorithmic rule ($\emptyset^{\mathrm{INF}}_{\bullet\Rightarrow}$) to infer $\Theta\,;\Gamma\,;\Xi \vDash N \bullet \Rightarrow \mathtt{nf}\,(N) \dashv \Xi\,;\cdot$. Let us show the required properties. Let us take an arbitrary $\Xi \vdash \widehat{\sigma}_0 : \mathtt{fav}(N)$ and $\Theta \vdash M$ such that $\Theta\,;\Gamma \vdash [\widehat{\sigma}_1]N \bullet \Rightarrow M$. Then we can take $\widehat{\sigma}_0$ as the required substitution:

1. $\Xi \vdash \widehat{\sigma}_0 : \mathtt{fav}(N) \cup \mathtt{fav}(\mathtt{nf}\,(N))$, since $\mathtt{fav}(\mathtt{nf}\,(N)) = \mathtt{fav}(N)$, and thus, $\mathtt{fav}(N) \cup \mathtt{fav}(\mathtt{nf}\,(N)) = \mathtt{fav}(N)$;
2. $\Xi \vdash \widehat{\sigma}_0 : \cdot$ vacuously;
3. $\Xi \vdash \widehat{\sigma}_0 \simeq^{\leqslant} \widehat{\sigma}_0 : \mathtt{fav}(N)$ by reflexivity;
4. Let us show $\Theta \vdash [\widehat{\sigma}_0]\mathtt{nf}\,(N) \simeq^{\leqslant} M$. Notice that $\Theta\,;\Gamma \vdash [\widehat{\sigma}_0]N \bullet \Rightarrow M$ can only be inferred by ($\emptyset^{\mathrm{INF}}_{\bullet\Rightarrow}$), and thus, $\Theta \vdash [\widehat{\sigma}_0]N \simeq^{\leqslant} M$. By Corollary 17, $\Theta \vdash [\widehat{\sigma}_0]N \simeq^{\leqslant} [\widehat{\sigma}_0]\mathtt{nf}\,(N)$, and then by transitivity, $\Theta \vdash [\widehat{\sigma}_0]\mathtt{nf}\,(N) \simeq^{\leqslant} M$, that is $\Theta \vdash [\widehat{\sigma}_0]\mathtt{nf}\,(N) \simeq^{\leqslant} M$.

■