Sound and Complete Bidirectional Typechecking for Higher-Rank Polymorphism with Existentials and Indexed Types

Joshua Dunfield
University of British Columbia
Vancouver, Canada

Neelakantan R. Krishnaswami
University of Birmingham
Birmingham, England

Abstract

Bidirectional typechecking, in which terms either synthesize a type or are checked against a known type, has become popular for its scalability, its error reporting, and its ease of implementation. Following principles from proof theory, bidirectional typing can be applied to many type constructs. The principles underlying a bidirectional approach to indexed types (*generalized algebraic datatypes*) are less clear. Building on proof-theoretic treatments of equality, we give a declarative specification of typing based on *focalization*. This approach permits declarative rules for coverage of pattern matching, as well as support for first-class existential types using a focalized subtyping judgment. We use refinement types to avoid explicitly passing equality proofs in our term syntax, making our calculus close to languages such as Haskell and OCaml. An explicit rule deduces when a type is principal, leading to reliable substitution principles for a rich type system with significant type inference. We also give a set of algorithmic typing rules, and prove that it is sound and complete with respect to the declarative system. The proof requires a number of technical innovations, including proving soundness and completeness in a mutually-recursive fashion.

1. Introduction

Consider a list type Vec with a numeric index representing its length, written in Haskell-like notation as follows:

```
data Vec : Nat -> * -> * where
  []   : A -> Vec 0 A
  (::) : A -> Vec n A -> Vec (succ n) A
```

We can use this definition to write a head function that always gives us an element of type A when the length is at least one:

$$\texttt{head} : \forall n, A.\ (\mathsf{Vec}\ \mathsf{succ}(n)\ A) \to A$$
$$\texttt{head}\ (x :: xs) = x$$

This clausal definition omits the clause for `[]`, which has an index of 0. The type annotation tells us that `head`'s argument has an index of $\mathsf{succ}(n)$ for some n. Since there is no natural number n such that $0 = \mathsf{succ}(n)$, the nil case cannot occur and can be omitted.

This is an entirely reasonable explanation for programmers, but language designers and implementors will have more questions. First, how can we implement such a type system? Clearly we needed some equality reasoning to justify leaving off the nil case, which is not trivial in general. Second, designers of functional languages are accustomed to the benefits of the Curry–Howard correspondence, and expect a *logical* reading of type systems to accompany the operational reading. So what is the logical reading of GADTs?

Since we relied on equality information to omit the nil case, it seems reasonable to look to logical accounts of equality. In proof theory, it is possible to formulate equality in (at least) two different ways. The better-known is the *identity type* of Martin-Löf, but GADTs actually correspond best to the equality of Schroeder-Heister (1994) and Girard (1992). The Girard–Schroeder-Heister (GSH) approach introduces equality via the reflexivity principle:

$$\frac{}{\Gamma \vdash t = t}$$

The GSH elimination rule was originally formulated in a sequent calculus style, as follows:

$$\frac{\text{for all } \theta.\ \text{if } \theta \in \mathsf{csu}(s,t) \text{ then } \theta(\Gamma) \vdash \theta(C)}{\Gamma, (s = t) \vdash C}$$

Here, we write $\mathsf{csu}(s,t)$ for a *complete set of unifiers* of s and t. So the rule says that we can eliminate an equality $s = t$ if, for every substitution θ that makes the terms s and t equal, we can give a proof of the goal C.

This rule has three important features, two good and one bad. First, the rule is an invertible left rule (the conclusion of the rule implies the premise, and it decomposes the assumptions to the left of the turnstile), which is known to correspond to a pattern matching rule (Krishnaswami 2009). This aligns with the use of GADTs in programming languages like Haskell and OCaml, which indeed use pattern matching to propagate equality information.

Second, when there are no unifiers, there are no premises: if we assume an inconsistent equation, we can immediately conclude the goal. Specializing the rule above to the equality $0 = 1$, we get:

$$\frac{}{\Gamma, (0 = 1) \vdash C}$$

Together, these two features line up nicely with our definition of `head`, where the impossibility of the case for `[]` was indicated by the *absence* of a pattern clause. So the use of equality in GADTs corresponds perfectly with the Girard–Schroeder-Heister equality.

Alas, we cannot simply give a proof term assignment for first-order logic and call it a day. The third important feature of the GSH equality rule is its use of unification: it works by treating the free variables of the two terms as unification variables. But type inference algorithms also use unification, introducing unification variables to stand for unknown types. So we need to understand how to integrate these two uses of unification, or at least how to keep them decently apart, in order to take this logical specification and implement type inference for it.

This problem—formulating indexed types in a logical style, while retaining the ability to do type inference for them—is the subject of this paper.

Contributions. The equivalence of GADTs to the combination of existential types and equality constraints has long been known

(Xi et al. 2003). Our fundamental contribution is to reduce GADTs to standard logical ingredients, *while retaining the implementability of the type system*. We accomplish this by formulating a system of indexed types in a bidirectional style (combining type *synthesis* with *checking* against a known type), which combines practical implementability with theoretical tidiness.

- Our language supports implicit higher-rank polymorphism (in which quantifiers can be nested under arrows) including existential types. While algorithms for higher-rank universal polymorphism are well-known (Peyton Jones et al. 2007; Dunfield and Krishnaswami 2013), our approach to supporting existential types is novel.

  Our system goes beyond the standard practice of tying existentials to datatype declarations (Läufer and Odersky 1994), in favour of a first-class treatment of implicit existential types. This approach has historically been thought difficult, because the unrestricted combination of universal and existential quantification seems to require mixed-prefix unification (i.e., solving equations under alternating quantifiers). We use the proof-theoretic technique of *focusing* to give a novel *polarized subtyping* judgment, which lets us treat alternating quantifiers in a way that retains decidability while maintaining essential subtyping properties, such as stability under substitution and transitivity.
- Our language includes equality types in the style of Girard and Schroeder-Heister, but without an explicit introduction form. for equality. Instead, we treat equalities as property types, in the style of intersection or refinement types: we do not write explicit equality proofs in our syntax, permitting us to more closely model how equalities are used in OCaml and Haskell.
- Our calculus includes nested pattern matching, which fits neatly in the bidirectional framework, and allows a formal specification of coverage checking with GADTs.
- Our declarative system tracks whether or not a derivation has a principal type. The system includes an unusual "higher-order principality" rule, which says that if only a single type can be synthesized for a term, then that type is principal. While this style of hypothetical reasoning is natural to explain to programmers, it is also extremely non-algorithmic.
- We formulate an algorithmic type system (Section 5) for our declarative calculus, and prove that typechecking is decidable, deterministic (5.3), and sound and complete (Sections 6–7) with respect to the declarative system.

Our algorithmic system (and, to a lesser extent, our declarative system) uses some techniques developed by Dunfield and Krishnaswami (2013), but we extend these to a far richer type language (existentials, indexed types, sums, products, equations over type variables), and we differ by supporting pattern matching, polarized subtyping, and principality tracking.

Appendix and proofs. The appendix has figures defining *all* the judgments, including some omitted here for space reasons. Full proofs are available from:

`github.com/joshuadunfield/lics39/raw/master/lics39_proofs.pdf`

2. Overview

To orient the reader, we give an overview and rationale of the novelties in our type system, before getting into the details of the typing rules and algorithm. As is well-known (Cheney and Hinze 2003; Xi et al. 2003), GADTs can be desugared into type expressions that use equality and existential types to express the return type constraints. These two features lead to all the key difficulties in typechecking for GADTs.

Universal, existentials, and type inference. Practical typed functional languages must support some degree of type inference, most critically the inference of type arguments. That is, if we have a function f of type $\forall a.\ a \to a$, and we want to apply it to the argument 3, then we want to write $f\ 3$, and not $f\ [\mathsf{Nat}]\ 3$ (as we would in pure System F). Even with a single type argument, the latter style is noisy, and programs using even moderate amounts of polymorphism rapidly become unreadable.

However, omitting type arguments has significant metatheoretical implications. In particular, it forces us to include subtyping in our typing rules, so that (for instance) the polymorphic type $\forall a.\ a \to a$ is a subtype of its instantiations (like $\mathsf{Nat} \to \mathsf{Nat}$).

For the subtype relation induced by polymorphism, subtype entailment is decidable (under modest restrictions). Matters get more complicated when *existential* types are added. Existentials are necessary to encode equality constraints in GADTs, but the naive combination of existential and universal types requires unification under a *mixed prefix* of alternating quantifiers (Miller 1992), which is undecidable. Thus, programming languages traditionally have stringently restricted the use of existential types. They tie existential introduction and elimination to datatype declarations, so that there is always a syntactic marker for when to introduce or eliminate existential types. This permits leaving existentials out of subtyping altogether, at the price of no longer permitting implicit subtyping (such as using $\lambda x.\ x + 1$ at type $\exists a.\ a \to a$).

While this is a practical solution, it increases the distance between surface languages and their type-theoretic cores. Our goal is to give a *direct* type-theoretic account of the features of our surface languages, avoiding complex elaboration passes. The key problem in mixed-prefix unification is that the order in which to instantiate quantifiers is unclear. When deciding $\Gamma \vdash \forall a.\ A(a) \leq \exists b.\ B(b)$, we have the choice to choose an instantiation for a or for b, so that we prove the subtype entailment $\Gamma \vdash A(t) \leq \exists b.\ B(b)$ or the subtype entailment $\Gamma \vdash \forall a.\ A(a) \leq B(t)$. An algorithm will introduce a unification variable for a and then for b, or the other way around—and this choice matters! With the first order, b may depend on a, but not vice versa; with the second order, the allowed dependencies are reversed. Accurate dependency tracking amounts to Skolemization, which means we have a "reduction" to the undecidable problem of higher-order unification.

We adopt an idea from polarized type theory. In the language of polarization, universals are a *negative* type, and existentials are a *positive* type. So we introduce two mutually-recursive subtype relations: $\Gamma \vdash A \leq^{+} B$ for positive types and $\Gamma \vdash A \leq^{-} B$ for negative types. The positive subtype relation only deconstructs existentials, and the negative subtype relation only deconstructs universals. This fixes the order in which quantifiers are instantiated, making the problem decidable (in fact, rather straightforward).

The price we pay is that fewer subtype entailments are derivable. Fortunately, all such entailments can be recovered by η-expansions. Moreover, the lost subtype entailments seem to all rely on "clever" quantifier reversals (which are rare in programming). So we keep fundamental expressivity, yet gain decidability.

Equality as a property. The usual convention in Haskell and OCaml is to make equality proofs in GADT definitions implicit. We would like to model this feature directly, so that our calculus stays close to surface languages, without sacrificing the logical reading of the system. In this case, the appropriate logical concepts come from the theory of intersection types. A typing judgment such as $e : A \times B$ can be viewed as giving instructions on how to construct a value (pair an A with a B). But types can also be viewed as *properties*, where $e : X$ is read "e has property X". To model

GADTs accurately, we treat equations $t = t'$ using a property type constructor $A \wedge P$, read "A with P", to model elements of type A satisfying the property (equation) P. We also introduce $P \supset A$, read "P implies A", for its adjoint dual. Then, standard rules for property types, which omit explicit proof terms, can explain why OCaml and Haskell do not require explicit equality proofs.

Handling equality constraints through intersection types also means that certain restrictions on typing that are useful for decidability, such as restricting property introduction to values, arise naturally from the semantic point of view—via the value restriction needed for soundly modeling intersection and union types (Davies and Pfenning 2000; Dunfield and Pfenning 2003).

Bidirectionality, pattern matching, and principality. Something that is not, by itself, novel in our approach is our decision to formulate both the declarative and algorithmic systems in a bidirectional style. Bidirectional checking (Pierce and Turner 2000) is a popular implementation choice for systems ranging from dependent types (Coquand 1996; Abel et al. 2008) and contextual types (Pientka 2008) to object-oriented languages (Odersky et al. 2001), but also has good proof-theoretic foundations (Watkins et al. 2004), making it useful both for specifying and implementing type systems. Bidirectional approaches make it clear to programmers where annotations are needed (which is good for specification), and can also remove unneeded nondeterminism from typing (which is good for both implementation and proving its correctness).

However, it is worth highlighting that because both bidirectionality and pattern matching arise from focalization, these two features fit together extremely well. In fact, by following the blueprint of focalization-based pattern matching, we can give a coverage-checking algorithm that explains when it is permissible to omit clauses in GADT pattern matching.

In the propositional case, the type synthesis judgment of a bidirectional type system generates principal types: if a type can be inferred for a term, that type is unique. This property is lost once quantifiers are introduced into the system, which is why it is not much remarked upon. However, prior work on GADTs, starting with Simonet and Pottier (2007), has emphasized the importance of the fact that handling equality constraints is much easier when the type of a scrutinee is principal. Essentially, this ensures that no existential variables can appear in equations, which prevents equation solving from interfering with unification-based type inference. The OutsideIn algorithm takes this consequence as a definition, permitting non-principal types just so long as they do not change the values of equations. However, Vytiniotis et al. (2011) note that while their system is sound, they no longer have a completeness result for their type system.

We use this insight to extend our bidirectional typechecking algorithm to track principality: The judgments we give track whether types are principal, and we use this to give a relatively simple specification for whether or not type annotations are needed. We are able to give a very natural spec to programmers—cases on GADTs must scrutinize terms with principal types, and an inferred type is principal just when it is the only type that can be inferred for that term—which soundly and completely corresponds to the implementation-side constraints: a type is principal when it contains no existential unification variables.

3. Examples

In this section, we give some examples of terms from our language, which illustrate the key features of our system and give a sense of how many type annotations are needed in practice. To help make this point clearly, all of the examples which follow are unsugared: they are the *actual* terms from our core calculus.

$$
\begin{array}{lll}
\text{Expressions} & e & ::= x \mid () \mid \lambda x.\, e \mid e\ s^+ \mid \mathsf{rec}\ x.\, v \mid (e : A) \\
& & \quad \mid \langle e_1, e_2 \rangle \mid \mathsf{inj}_1\ e \mid \mathsf{inj}_2\ e \mid \mathsf{case}(e, \Pi) \\
& & \quad \mid [] \mid e_1 :: e_2 \\
\text{Values} & v & ::= x \mid () \mid \lambda x.\, e \mid \mathsf{rec}\ x.\, v \mid (v : A) \\
& & \quad \mid \langle v_1, v_2 \rangle \mid \mathsf{inj}_1\ v \mid \mathsf{inj}_2\ v \mid [] \mid v_1 :: v_2 \\
\text{Spines} & s & ::= \cdot \mid e\ s \\
\text{Nonempty spines} & s^+ & ::= e\ s \\
\text{Patterns} & \rho & ::= x \mid \langle \rho_1, \rho_2 \rangle \mid \mathsf{inj}_1\ \rho \mid \mathsf{inj}_2\ \rho \mid [] \mid \rho_1 :: \rho_2 \\
\text{Branches} & \pi & ::= \vec{\rho} \Rightarrow e \\
\text{Branch lists} & \Pi & ::= \cdot \mid (\pi \mid \Pi)
\end{array}
$$

Figure 1. Source syntax

Mapping over lists. First, we begin with the traditional *map* function, which takes a function and applies it to every element of a list.

$$
\begin{array}{l}
\mathsf{rec}\ \mathit{map}.\, \lambda f.\, \lambda xs.\, \mathsf{case}\big(xs,\quad [] \Rightarrow [] \\
\qquad\qquad\qquad\qquad \mid y :: ys \Rightarrow (f\ y) :: \mathit{map}\ f\ ys\big) \\
: \forall n : \mathbb{N}.\, \forall \alpha : \star.\, \forall \beta : \star.\, (\alpha \to \beta) \to \mathsf{Vec}\ n\ \alpha \to \mathsf{Vec}\ n\ \beta
\end{array}
$$

This code is simply a recursive function that case-analyzes its second argument xs. Given an empty xs, it returns the empty list; given a cons cell $y :: ys$, it applies the argument function f to the head y and making a recursive call on the tail ys.

In addition, we annotate the definition with a type. We have two type parameters α and β for the input and output element types. Since we are working with length-indexed lists, we also have a length index parameter n, which lets us show by typing that the input and output to *map* have the same length.

In our system, this type annotation is mandatory. Full type inference for definitions using GADTs requires polymorphic recursion, which is undecidable. As a result, this example also requires annotation in OCaml and GHC Haskell. However, Haskell and OCaml infer polymorphic types when no polymorphic recursion is needed. We adopt the simpler rule that *all* polymorphic definitions are annotated. This choice is motivated by Vytiniotis et al. (2010), who analyzed a large corpus of Haskell code and showed that implicit let-generalization was rarely used: programmers tend to annotate polymorphic definitions for documentation purposes.

Nested patterns and GADTs. Now, we consider the `zip` function, which converts a pair of lists into a list of pairs. In ordinary ML or Haskell, we must consider what to do when the two lists are not the same length. However, with length-indexed lists, we can statically reject passing two lists of differing length:

$$
\begin{array}{l}
\mathsf{rec}\ \mathit{zip}.\, \lambda p.\, \mathsf{case}\big(p,\quad ([], []) \Rightarrow [] \\
\qquad\qquad\qquad\qquad \mid (x :: xs, y :: ys) \Rightarrow (x, y) :: \mathit{zip}\ (xs, ys)\big) \\
: \forall n : \mathbb{N}.\, \forall \alpha : \star.\, \forall \beta : \star.\, (\mathsf{Vec}\ n\ \alpha \times \mathsf{Vec}\ n\ \beta) \to \mathsf{Vec}\ n\ (\alpha \times \beta)
\end{array}
$$

In this case expression, we give only two patterns, one for when both lists are empty and one for when both lists have elements, with the type annotation indicating that both lists must be of length n. Typing shows that the cases where one list is empty and the other non-empty are impossible, so our coverage checking rules accept this as a complete set of patterns. This example also illustrates that we support nested pattern matching.

4. Declarative Typing

Expressions. Expressions (Figure 1) are variables x; the unit value $()$; functions $\lambda x.\, e$; applications to a spine $e\ s^+$; fixed points $\mathsf{rec}\ x.\, v$; annotations $(e : A)$; pairs $\langle e_1, e_2 \rangle$; injections into a sum type $\mathsf{inj}_k\ e$; case expressions $\mathsf{case}(e, \Pi)$ where Π is a list of branches π, which can eliminate pairs and injections (see below); the empty vector $[]$; and consing a head e_1 to a tail vector e_2.

Universal variables	α, β, γ	
Sorts	κ	$::= \star \mid \mathbb{N}$
Types	A, B, C	$::= 1 \mid A \to B \mid A + B \mid A \times B$ $\mid \alpha \mid \forall \alpha : \kappa.\ A \mid \exists \alpha : \kappa.\ A$ $\mid P \supset A \mid A \wedge P \mid \mathsf{Vec}\ t\ A$
Terms/monotypes	t, τ, σ	$::= \mathsf{zero} \mid \mathsf{succ}(t) \mid 1 \mid \alpha$ $\mid \tau \to \sigma \mid \tau + \sigma \mid \tau \times \sigma$
Propositions	P, Q	$::= t = t'$
Contexts	Ψ	$::= \cdot \mid \Psi, \alpha : \kappa \mid \Psi, x : A\ p$
Polarities	$\pm$	$::= + \mid -$
Binary connectives	$\oplus$	$::= \to \mid + \mid \times$
Principalities	p, q	$::= \mathbf{!} \mid \underbrace{\not!}_{\text{sometimes omitted}}$

Figure 2. Syntax of declarative types and contexts

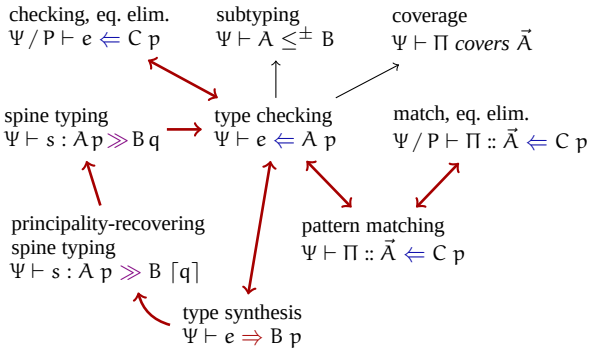

Figure 3. Dependency structure of the declarative judgments

$\boxed{\Psi \vdash A \leq^{\pm} B}$ Under context Ψ, type A is a subtype of B, decomposing head connectives of polarity $\pm$

$$\frac{\Psi \vdash A\ \mathit{type} \qquad \mathit{nonpos}(A) \qquad \mathit{nonneg}(A)}{\Psi \vdash A \leq^{\pm} A}\ \leq\mathsf{Refl}\pm$$

$$\frac{\Psi \vdash A \leq^{-} B \qquad \mathit{nonpos}(A) \qquad \mathit{nonpos}(B)}{\Psi \vdash A \leq^{+} B}\ \leq^{-}_{+}$$

$$\frac{\Psi \vdash A \leq^{+} B \qquad \mathit{nonneg}(A) \qquad \mathit{nonneg}(B)}{\Psi \vdash A \leq^{-} B}\ \leq^{+}_{-}$$

$$\frac{\Psi \vdash \tau : \kappa \qquad \Psi \vdash [\tau/\alpha]A \leq^{-} B}{\Psi \vdash \forall \alpha{:}\kappa.\ A \leq^{-} B}\ \leq\forall\mathsf{L} \qquad \frac{\Psi, \beta : \kappa \vdash A \leq^{-} B}{\Psi \vdash A \leq^{-} \forall \beta{:}\kappa.\ B}\ \leq\forall\mathsf{R}$$

$$\frac{\Psi, \alpha : \kappa \vdash A \leq^{+} B}{\Psi \vdash \exists \alpha{:}\kappa.\ A \leq^{+} B}\ \leq\exists\mathsf{L} \qquad \frac{\Psi \vdash \tau : \kappa \qquad \Psi \vdash A \leq^{+} [\tau/\beta]B}{\Psi \vdash A \leq^{+} \exists \beta{:}\kappa.\ B}\ \leq\exists\mathsf{R}$$

Figure 4. Subtyping in the declarative system

Values v are standard for a call-by-value semantics; the variables introduced by fixed points are considered values, because we only allow fixed points of values. A spine s is a list of expressions—arguments to a function. Allowing empty spines (written $\cdot$) is convenient in the typing rules, but would be strange in the source syntax, so (in the grammar of expressions e) we require a nonempty spine s^+. We usually omit the empty spine $\cdot$, writing $e_1\ e_2$ instead of $e_1\ e_2\ \cdot$. Since we use juxtaposition for both application $e\ s^+$ and spines, some strings are ambiguous; we resolve this ambiguity in favour of the spine, so $e_1\ e_2\ e_3$ is parsed as the application of e_1 to the spine $e_2\ e_3$, which is technically $e_2\ (e_3\ \cdot)$. Patterns ρ consist of pattern variables, pairs, and injections. A branch π is a *sequence* of patterns $\vec{\rho}$ with a branch body e. We represent patterns as sequences, which enables us to deconstruct tuple patterns.

Types. We write types as A, B and C. We have the unit type 1, functions $A \to B$, sums $A + B$, and products $A \times B$. We have universal and existential types $\forall \alpha : \kappa.\ A$ and $\exists \alpha : \kappa.\ A$; these are predicative quantifiers over monotypes (see below). We write α, β, etc. for type variables; these are universal, except when bound within an existential type. We also have a *guarded type* $P \supset A$, read "P implies A". This implication corresponds to type A, provided P holds. Its dual is the *asserting type* $A \wedge P$, read "A with P", which witnesses the proposition P. In both, P has no runtime content.

Sorts, terms, monotypes, and propositions. Terms and monotypes t, τ, σ share a grammar but are distinguished by their *sorts* κ. Natural numbers zero and $\mathsf{succ}(t)$ are *terms* and have sort $\mathbb{N}$. Unit 1 has the sort $\star$ of *monotypes*. A variable α stands for a term or a monotype, depending on the sort κ annotating its binder. Functions, sums, and products of monotypes are monotypes and have sort $\star$. We tend to prefer t for terms and σ, τ for monotypes.

A proposition P or Q is simply an equation $t = t'$. Note that terms, which represent runtime-irrelevant information, are distinct from expressions; however, an expression may include type annotations of the form $P \supset A$ and $A \wedge P$, where P contains terms.

Contexts. A declarative context Ψ is an *ordered* sequence of universal variable declarations $\alpha : \kappa$ and expression variable typings $x : A\ p$, where p denotes whether the type A is principal (Section 4.2). A variable α can be free in a type A only if α was declared to the left: $\alpha : \star$, $x : \alpha\ p$ is well-formed, but $x : \alpha\ p$, $\alpha : \star$ is not.

4.1 Subtyping

We give our two subtyping relations in Figure 4. We treat the universal quantifier as a negative type (since it is a function in System F), and the existential as a positive type (since it is a pair in System F). We have two typing rules for each of these connectives, corresponding to the left and right rules for universals and existentials in the sequent calculus. We treat all other types as having no polarity. The positive and negative subtype judgments are mutually recursive, and the $\leq^{-}_{+}$ rule permits switching the polarity of subtyping from positive to negative when both of the types are non-positive, and conversely for $\leq^{+}_{-}$. When both types are neither positive nor negative, we require them to be equal ($\leq\mathsf{Refl}$).

In logical terms, functions and guarded types are negative; sums, products and assertion types are positive. We could potentially operate on these types in the negative and positive subtype relations, respectively. Leaving out (for example) function subtyping means that we will have to do some η-expansions to get programs to typecheck; we omit these rules to keep the implementation complexity low. This also illustrates a nice feature of bidirectional typing: we are relatively free to adjust the subtype relation to taste.

4.2 Typing judgments

Principality. Our typing judgments carry *principalities*: $A\ \mathbf{!}$ means that A is principal, and $A\ \not!$ means A is not principal. Note that a principality is part of a judgment, not part of a type. In the checking judgment $\Psi \vdash e \Leftarrow A\ p$ the type A is input; if $p = \mathbf{!}$, we know that e is not the result of guessing. For example, the e in $(e : A)$ is checked against $A\ \mathbf{!}$. In the synthesis judgment $\Psi \vdash e \Rightarrow A\ p$, the type A is output, and $p = \mathbf{!}$ means it is impossible to synthesize any other type, as in $\Psi \vdash (e : A) \Rightarrow A\ \mathbf{!}$.

We sometimes omit a principality when it is $\not!$ ("not principal"). We write $p \sqsubseteq q$, read "p at least as principal as q", for the reflexive closure of $\mathbf{!} \sqsubseteq \not!$.

$\boxed{e\ \textit{chk-I}}$ Expression e is a checked introduction form $\quad \overline{\lambda x.\, e\ \textit{chk-I}} \quad \overline{()\ \textit{chk-I}} \quad \overline{\langle e_1, e_2\rangle\ \textit{chk-I}} \quad \overline{\mathsf{inj}_k\ e\ \textit{chk-I}} \quad \overline{[]\ \textit{chk-I}} \quad \overline{e_1 :: e_2\ \textit{chk-I}}$

Figure 5. "Checking intro form"

$\boxed{\Psi \vdash e \Leftarrow A\ p}$ Under context Ψ, expression e checks against input type A

$\boxed{\Psi \vdash e \Rightarrow A\ p}$ Under context Ψ, expression e synthesizes output type A

$\boxed{\Psi \vdash s : A\ p \gg C\ q \quad \Psi \vdash s : A\ p \gg C\ \lceil q \rceil}$ Under context Ψ, passing spine s to a function of type A synthesizes type C; in the $\lceil q \rceil$ form, recover principality in q if possible

$\boxed{\Psi \vdash P\ \textit{true}}$ Under context Ψ, check P

$$\frac{}{\Psi \vdash (t = t)\ \textit{true}}\ \mathsf{DeclCheckpropEq}$$

$$\frac{x : A\ p \in \Psi}{\Psi \vdash x \Rightarrow A\ p}\ \mathsf{DeclVar} \qquad \frac{\Psi \vdash e \Rightarrow A\ q \qquad \Psi \vdash A \leq^{\mathrm{pol}(B)} B}{\Psi \vdash e \Leftarrow B\ p}\ \mathsf{DeclSub} \qquad \frac{\Psi \vdash A\ \textit{type} \qquad \Psi \vdash e \Leftarrow A\ !}{\Psi \vdash (e : A) \Rightarrow A\ !}\ \mathsf{DeclAnno}$$

$$\frac{}{\Psi \vdash () \Leftarrow 1\ p}\ \mathsf{Decl1I} \qquad \frac{\nu\ \textit{chk-I} \qquad \Psi, \alpha : \kappa \vdash \nu \Leftarrow A\ p}{\Psi \vdash \nu \Leftarrow (\forall \alpha : \kappa.\, A)\ p}\ \mathsf{Decl}\forall\mathsf{I} \qquad \frac{\Psi \vdash \tau : \kappa \qquad \Psi \vdash e\ s : [\tau/\alpha]A \not\!/\!\!/ \gg C\ q}{\Psi \vdash e\ s : (\forall \alpha : \kappa.\, A)\ p \gg C\ q}\ \mathsf{Decl}\forall\mathsf{Spine}$$

$$\frac{\Psi \vdash P\ \textit{true} \qquad \Psi \vdash e \Leftarrow A\ p}{\Psi \vdash e \Leftarrow (A \wedge P)\ p}\ \mathsf{Decl}\wedge\mathsf{I} \qquad \frac{\nu\ \textit{chk-I} \qquad \Psi\ /\ P \vdash \nu \Leftarrow A\ !}{\Psi \vdash \nu \Leftarrow (P \supset A)\ !}\ \mathsf{Decl}\supset\mathsf{I} \qquad \frac{\Psi \vdash P\ \textit{true} \qquad \Psi \vdash e\ s : A\ p \gg C\ q}{\Psi \vdash e\ s : (P \supset A)\ p \gg C\ q}\ \mathsf{Decl}\supset\mathsf{Spine}$$

$$\frac{\Psi, x : A\ p \vdash \nu \Leftarrow A\ p}{\Psi \vdash \mathsf{rec}\ x.\, \nu \Leftarrow A\ p}\ \mathsf{DeclRec} \qquad \frac{\Psi, x : A\ p \vdash e \Leftarrow B\ p}{\Psi \vdash \lambda x.\, e \Leftarrow A \to B\ p}\ \mathsf{Decl}\to\mathsf{I} \qquad \frac{\Psi \vdash e \Rightarrow A\ p \qquad \Psi \vdash s : A\ p \gg C\ \lceil q \rceil}{\Psi \vdash e\ s \Rightarrow C\ q}\ \mathsf{Decl}\to\mathsf{E}$$

$$\frac{\Psi \vdash s : A\ ! \gg C \not\!/\!\!/ \qquad \text{for all } C'.\ \text{if } \Psi \vdash s : A\ ! \gg C' \not\!/\!\!/ \text{ then } C' = C}{\Psi \vdash s : A\ ! \gg C\ \lceil ! \rceil}\ \mathsf{DeclSpineRecover} \qquad \frac{\Psi \vdash s : A\ p \gg C\ q}{\Psi \vdash s : A\ p \gg C\ \lceil q \rceil}\ \mathsf{DeclSpinePass}$$

$$\frac{}{\Psi \vdash \cdot : A\ p \gg A\ p}\ \mathsf{DeclEmptySpine} \qquad \frac{\Psi \vdash e \Leftarrow A\ p \qquad \Psi \vdash s : B\ p \gg C\ q}{\Psi \vdash e\ s : A \to B\ p \gg C\ q}\ \mathsf{Decl}\to\mathsf{Spine} \qquad \frac{\Psi \vdash e \Rightarrow A\ ! \qquad \Psi \vdash \Pi :: A \Leftarrow C\ p \qquad \Psi \vdash \Pi\ \textit{covers}\ A}{\Psi \vdash \mathsf{case}(e, \Pi) \Leftarrow C\ p}\ \mathsf{DeclCase}$$

$\boxed{\Psi\ /\ P \vdash e \Leftarrow C\ p}$ Under context Ψ, incorporate proposition P and check e against C

$$\frac{\mathsf{mgu}(\sigma, \tau) = \bot}{\Psi\ /\ (\sigma = \tau) \vdash e \Leftarrow C\ p}\ \mathsf{DeclCheck}\bot \qquad \frac{\mathsf{mgu}(\sigma, \tau) = \theta \qquad \theta(\Psi) \vdash \theta(e) \Leftarrow \theta(C)\ p}{\Psi\ /\ (\sigma = \tau) \vdash e \Leftarrow C\ p}\ \mathsf{DeclCheckUnify}$$

Figure 6. Declarative typing, omitting rules for $\times$, $+$, and Vec

Spine judgments. The ordinary form of spine judgment, $\Psi \vdash s : A\ p \gg C\ q$, says that if arguments s are passed to a function of type A, the function returns type C. For a function e applied to one argument e_1, we write $e\ e_1$ as syntactic sugar for $e\ (e_1\ \cdot)$. Supposing e synthesizes $A_1 \to A_2$, we apply Decl→Spine, checking e_1 against A_1 and using DeclEmptySpine to derive $\Psi \vdash \cdot : A_2\ p \gg A_2\ p$.

Rule Decl∀Spine does not decompose $e\ s$ but instantiates a $\forall$. Note that, even if the given type $\forall \alpha : \kappa.\, A$ is principal ($p = !$), the type $[\tau/\alpha]A$ in the premise is not principal—we could choose a different τ. In fact, the q in Decl∀Spine is also always $\not\!/\!\!/$, because no rule deriving the ordinary spine judgment can recover principality.

The *recovery spine judgment* $\Psi \vdash s : A\ p \gg C\ \lceil q \rceil$, however, can restore principality in situations where the choice of τ in Decl∀Spine cannot affect the result type C. If A is principal ($p = !$) but the ordinary spine judgment produces a non-principal C, we can try to recover principality with DeclSpineRecover. Its first premise is $\Psi \vdash s : A\ ! \gg C \not\!/\!\!/$; its second premise (really, an infinite set of premises) quantifies over all derivations of $\Psi \vdash s : A\ ! \gg C' \not\!/\!\!/$. If $C' = C$ in all such derivations, then the ordinary spine rules erred on the side of caution: C is actually principal, so we can set $q = !$ in the conclusion of DeclSpineRecover.

If some $C' \neq C$, then C is certainly not principal, and we must apply DeclSpinePass, which simply transitions from the ordinary judgment to the recovery judgment.

Figure 3 shows the dependencies between the declarative judgments. Given the cycle containing the spine typing judgments, we need to stop and ask: Is DeclSpineRecover well-founded? For well-foundedness of type systems, we can often make a straightforward argument that, as we move from the conclusion of a rule to its premises, either the expression gets smaller, or the expression stays the same but the type gets smaller. In DeclSpineRecover, neither the expression nor the type get smaller. Fortunately, the rule that gives rise to the arrow from "spine typing" to "type checking" in Figure 3—Decl→Spine—*does* decompose its subject, and any derivations of a recovery judgment lurking within the second premise of DeclSpineRecover must be for a smaller spine. In the appendix (Lemma **??**, p. **??**), we prove that the recovery judgment, and all the other declarative judgments, are well-founded.

Example. Appendix A has an example showing how the spine typing rules work to recover principality.

Subtyping. Rule DeclSub invokes the subtyping judgment, at the polarity of B, the type being checked against. This allows DeclSub to play the role of an existential introduction rule, by applying subtyping rule $\leq\exists R$ when B is an existential type.

Pattern matching. Rule DeclCase checks that the scrutinee has a principal type, and then invokes the two main judgments for pattern matching. The $\Psi \vdash \Pi :: \vec{A} \Leftarrow C\ p$ judgment checks that each branch in the list of branches Π is well-typed, and the $\Psi \vdash \Pi$ *covers* $\vec{A}$ judgment does coverage checking for the list of branches. Both of these judgments take a vector $\vec{A}$ of pattern types to simplify the specification of coverage checking.

The $\Psi \vdash \Pi :: \vec{A} \Leftarrow C\ p$ judgment (rules in the appendix's Figure 15) systematically checks the coverage of each branch in Π: rule DeclMatchEmpty succeeds on the empty list, and DeclMatchSeq checks one branch and recurs on the remaining branches. Rules for sums, units, and products break down patterns left to right, one constructor at a time. Products also extend the sequences of patterns and types, with DeclMatch$\times$ breaking down a pattern vector headed by a pair pattern $\langle p, p' \rangle, \vec{p}$ into $p, p', \vec{p}$, and breaking down the type sequence $(A \times B), \vec{C}$ into $A, B, \vec{C}$. Once all the patterns are eliminated, the DeclMatchBase rule says that if the body typechecks, then the branch typechecks. For completeness, the variable and wildcard rules are restricted so that any top-level existentials and equations are eliminated before discarding the type.

The existential elimination rule DeclMatch$\exists$ unpacks an existential type, and DeclMatch$\wedge$ breaks apart a conjunction by eliminating the equality using unification. The DeclMatch$\bot$ rule says that if the equation is false then the branch always succeeds, because this case is impossible. The DeclMatchUnify rule unifies the two terms of an equation and applies the substitution before continuing to check typing. Together, these two rules implement the Schroeder-Heister equality elimination rule. Because our language of terms has only simple first-order terms, either unification will fail, or there is a most general unifier.

The $\Psi \vdash \Pi$ *covers* $\vec{A}$ judgment (in the appendix's Figure 16) checks whether a set of patterns covers all possible cases. As with match typing, we systematically deconstruct the sequence of types in the branch, but we also need auxiliary operations to *expand* the patterns. For example, the $\Pi \overset{\times}{\rightsquigarrow} \Pi'$ operation takes every branch $\langle p, p' \rangle, \vec{\rho} \Rightarrow e$ and expands it to $p, p', \vec{\rho} \Rightarrow e$. To keep the sequence of patterns aligned with the sequence of types, we also expand variables and wildcard patterns into two wildcards: $x, \vec{\rho} \Rightarrow e$ becomes $\_, \_, \vec{\rho} \Rightarrow e$. After expanding out all the pairs, DeclCovers$\times$ checks coverage by breaking down the pair type.

For sum types, we expand a list of branches into *two* lists, one for each injection. So $\Pi \overset{+}{\rightsquigarrow} \Pi_L \parallel \Pi_R$ will send all branches headed by $\mathsf{inj}_1\ p$ into Π_L and all branches headed by $\mathsf{inj}_2\ p$ into Π_R, with variables and wildcards being sent to both sides. Then DeclCovers$+$ checks the left and right branches independently.

As with typing, DeclCovers$\exists$ just unpacks the existential type. Likewise, DeclCoversEqBot and DeclCoversEq handle the two cases arising from equations. If an equation is unsatisfiable, coverage succeeds since there are no possible values of that type. If it is satisfiable, we apply the substitution and continue coverage checking.

These rules do not check for redundancy: DeclCoversEmpty applies even when branches are left over. When DeclCoversEmpty is applied, we could mark the $\cdot \Rightarrow e_1$ branch, and issue a warning for unmarked branches. This seems better as a warning than an error, since redundancy is not stable under substitution. For example, a case over $(\mathsf{Vec}\ n\ A)$ with [] and :: branches is not redundant—but if we substitute 0 for n, the :: branch becomes redundant.

Synthesis. Bidirectional typing is a form of partial type inference, which Pierce and Turner (2000) said should "eliminate especially those type annotations that are both *common* and *silly*". But our rules are rather parsimonious in what they synthesize; for instance, () does not synthesize 1, and so might need an annotation. Fortunately, it would be straightforward to add such rules, following the style of Dunfield and Krishnaswami (2013).

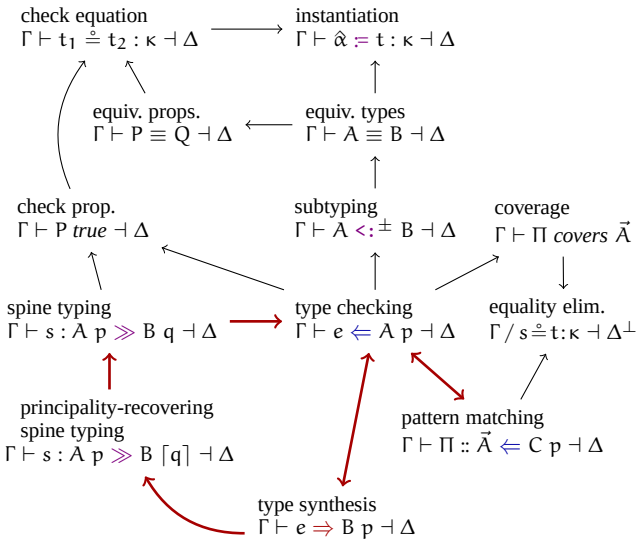

Figure 7. Dependency structure of the algorithmic judgments

5. Algorithmic Typing

Our algorithmic rules closely mimic our declarative rules, except that whenever a declarative rule would make a guess, the algorithmic rule adds to the context an existential variable (written with a hat $\hat{\alpha}$). As typechecking proceeds, we add solutions to the existential variables, reflecting increasing knowledge. Hence, each declarative typing judgment has a corresponding algorithmic judgment with an output context as well as an input context. The algorithmic type checking judgment $\Gamma \vdash e \Leftarrow A\ p \dashv \Delta$ takes an input context Γ and yields an output context Δ that includes increased knowledge about what the types have to be. The notion of increasing knowledge is formalized by a judgment $\Gamma \longrightarrow \Delta$ (Section 5.2).

Figure 7 shows a dependency graph of the algorithmic judgments. Each declarative judgment has a corresponding algorithmic judgment, but the algorithmic system adds judgments such as type equivalence checking $\Gamma \vdash A \equiv B \dashv \Delta$ and variable instantiation $\Gamma \vdash \hat{\alpha} := t : \kappa \dashv \Delta$. Declaratively, these judgments correspond to uses of reflexivity axioms; algorithmically, they correspond to solving existential variables to equate terms.

We give the algorithmic typing rules in Figure 11; rules for most other judgments are in the appendix. Our style of specification broadly follows Dunfield and Krishnaswami (2013): we adapt their mechanisms of variable instantiation, context extension, and context application (to both types and other contexts). Our versions of these mechanisms, however, support indices, equations over universal variables, and the $\exists/\supset/\wedge$ connectives. We also differ in our formulation of spine typing, and by being able to track which types are principal. The example in Appendix A shows how the algorithmic spine typing rules work to recover principality.

Syntax. Expressions are the same as in the declarative system.

Existential variables. The algorithmic system adds existential variables $\hat{\alpha}, \hat{\beta}, \hat{\gamma}$ to types and terms/monotypes (Figure 8). We use the same meta-variables $A, \ldots$ for algorithmic types. We write u for either a universal variable α or an existential variable $\hat{\alpha}$.

Contexts. An algorithmic context Γ is a sequence that, like a declarative context, may contain universal variable declarations

$$
\begin{array}{lrcl}
\text{Universal variables} & \alpha, \beta, \gamma & & \\
\text{Existential variables} & \hat{\alpha}, \hat{\beta}, \hat{\gamma} & & \\
\text{Variables} & u & ::= & \alpha \mid \hat{\alpha} \\
\text{Types} & A, B, C & ::= & 1 \mid A \to B \mid A + B \mid A \times B \\
 & & & \mid \alpha \mid \hat{\alpha} \mid \forall \alpha : \kappa.\, A \mid \exists \alpha : \kappa.\, A \\
 & & & \mid P \supset A \mid A \wedge P \mid \mathsf{Vec}\ t\ A \\
\text{Propositions} & P, Q & ::= & t = t' \\
\text{Binary connectives} & \oplus & ::= & \to \mid + \mid \times \\
\text{Terms/monotypes} & t, \tau, \sigma & ::= & \mathsf{zero} \mid \mathsf{succ}(t) \mid 1 \mid \alpha \mid \hat{\alpha} \\
 & & & \mid \tau \to \sigma \mid \tau + \sigma \mid \tau \times \sigma \\
\text{Contexts} & \Gamma, \Delta, \Theta & ::= & \cdot \mid \Gamma, u : \kappa \mid \Gamma, x : A\ p \\
 & & & \mid \Gamma, \hat{\alpha} : \kappa = \tau \mid \Gamma, \alpha = t \mid \Gamma, \blacktriangleright_u \\
\text{Complete contexts} & \Omega & ::= & \cdot \mid \Omega, \alpha : \kappa \mid \Omega, x : A\ p \\
 & & & \mid \Omega, \hat{\alpha} : \kappa = \tau \mid \Omega, \alpha = t \mid \Omega, \blacktriangleright_u \\
\text{Possibly-inconsistent contexts} & \Delta^{\bot} & ::= & \Delta \mid \bot
\end{array}
$$

Figure 8. Syntax of types, contexts, and other objects in the algorithmic system

$$
[\Gamma]\alpha = \begin{cases} [\Gamma]\tau & \text{when } (\alpha = \tau) \in \Gamma \\ \alpha & \text{otherwise} \end{cases}
$$

$$
\begin{array}{lcllcl}
[\Gamma](P \supset A) & = & ([\Gamma]P) \supset ([\Gamma]A) & [\Gamma](\forall \alpha : \kappa.\, A) & = & \forall \alpha : \kappa.\, [\Gamma]A \\
[\Gamma](A \wedge P) & = & ([\Gamma]A) \wedge ([\Gamma]P) & [\Gamma](\exists \alpha : \kappa.\, A) & = & \exists \alpha : \kappa.\, [\Gamma]A \\
[\Gamma](A \oplus B) & = & ([\Gamma]A) \oplus ([\Gamma]B) & & & \\
[\Gamma](\mathsf{Vec}\ t\ A) & = & \mathsf{Vec}\ ([\Gamma]t)\ ([\Gamma]A) & [\Gamma](t_1 = t_2) & = & ([\Gamma]t_1) = ([\Gamma]t_2)
\end{array}
$$

$$
\begin{array}{lcl}
\left[\Gamma[\hat{\alpha} : \kappa = \tau]\right]\hat{\alpha} & = & [\Gamma]\tau \\
\left[\Gamma[\hat{\alpha} : \kappa]\right]\hat{\alpha} & = & \hat{\alpha}
\end{array}
$$

Figure 9. Applying a context, as a substitution, to a type

$\alpha : \kappa$ and expression variable typings $x : A\ p$. However, it may also have (1) *unsolved* existential variable declarations $\hat{\alpha} : \kappa$ (included in the $\Gamma, u : \kappa$ production); (2) *solved* existential variable declarations $\hat{\alpha} : \kappa = \tau$; (3) *equations* over universal variables $\alpha = \tau$; and (4) *markers* $\blacktriangleright_u$.

An equation $\alpha = \tau$ must appear to the right of the universal variable's declaration $\alpha : \kappa$. We use markers as delimiters within contexts. For example, rule $\supset I$ adds $\blacktriangleright_P$, which tells it how much of its last premise's output context $(\Delta, \blacktriangleright_P, \Delta')$ should be dropped. (We abuse notation by writing $\blacktriangleright_P$ rather than cluttering the context with a dummy α and writing $\blacktriangleright_\alpha$.)

A complete algorithmic context, denoted by Ω, is an algorithmic context with no unsolved existential variable declarations.

Assuming an equality can yield inconsistency: for example, $\mathsf{zero} = \mathsf{succ}(\mathsf{zero})$. We write $\Delta^{\bot}$ for either a valid algorithmic context Δ or inconsistency $\bot$.

5.1 Context substitution $[\Gamma]A$ and hole notation $\Gamma[\Theta]$

An algorithmic context can be viewed as a substitution for its solved existential variables. For example, $\hat{\alpha} = 1, \hat{\beta} = \hat{\alpha} \to 1$ can be applied as if it were the substitution $1/\hat{\alpha}, (\hat{\alpha} \to 1)/\hat{\beta}$ (applied right to left), or the simultaneous substitution $1/\hat{\alpha}, (1 \to 1)/\hat{\beta}$. We write $[\Gamma]A$ for Γ applied as a substitution (Figure 9).

Applying a complete context to a type A (provided it is well-formed: $\Omega \vdash A$ *type*) yields a type $[\Omega]A$ with no existentials. Such a type is well-formed under the *declarative* context obtained by dropping all the existential declarations and applying Ω to declarations $x : A$ (to yield $x : [\Omega]A$). We can think of this context as the result of applying Ω to itself: $[\Omega]\Omega$. More generally, we can apply Ω to any context Γ that it extends: context application $[\Omega]\Gamma$ is given in Figure 10. The application $[\Omega]\Gamma$ is defined if and only if $\Gamma \longrightarrow \Omega$ (context extension; see Section 5.2), and applying Ω to any such Γ yields the same declarative context $[\Omega]\Omega$.

In addition to appending declarations (as in the declarative system), we sometimes insert and replace declarations, so a notation

$$
\begin{array}{lcl}
[\cdot]\cdot & = & \cdot \\
[\Omega, x : A\ p](\Gamma, x : A_\Gamma\ p) & = & [\Omega]\Gamma,\ x : [\Omega]A\ p \ \text{ if } [\Omega]A = [\Omega]A_\Gamma \\
[\Omega, \alpha : \kappa](\Gamma, \alpha : \kappa) & = & [\Omega]\Gamma,\ \alpha : \kappa \\
[\Omega, \blacktriangleright_u](\Gamma, \blacktriangleright_u) & = & [\Omega]\Gamma \\
[\Omega, \alpha = t](\Gamma, \alpha = t') & = & \left[[\Omega]t/\alpha\right][\Omega]\Gamma \quad \text{if } [\Omega]t = [\Omega]t' \\
[\Omega, \hat{\alpha} : \kappa = t]\Gamma & = & \begin{cases} [\Omega]\Gamma' \text{ when } \Gamma = (\Gamma', \hat{\alpha} : \kappa = t') \\ [\Omega]\Gamma' \text{ when } \Gamma = (\Gamma', \hat{\alpha} : \kappa) \\ [\Omega]\Gamma \quad \text{otherwise} \end{cases}
\end{array}
$$

Figure 10. Applying a complete context Ω to a context

for contexts with a hole is useful: $\Gamma = \Gamma_0[\Theta]$ means Γ has the form $(\Gamma_L, \Theta, \Gamma_R)$. For example, if $\Gamma = \Gamma_0[\hat{\beta}] = (\hat{\alpha}, \hat{\beta}, x : \hat{\beta})$, then $\Gamma_0[\hat{\beta} = \hat{\alpha}] = (\hat{\alpha}, \hat{\beta} = \hat{\alpha}, x : \hat{\beta})$.

5.2 The context extension relation $\Gamma \longrightarrow \Delta$

A context Γ *is extended by* a context Δ, written $\Gamma \longrightarrow \Delta$, if Δ has at least as much information as Γ, while conforming to the same declarative context—that is, $[\Omega]\Gamma = [\Omega]\Delta$ for some Ω. In a sense, $\Gamma \longrightarrow \Delta$ says that Γ is *entailed by* Δ: all positive information derivable from Γ can also be derived from Δ (which may have more information, say, that $\hat{\alpha}$ is equal to a particular type). We give the rules for extension in Figure 14 in the appendix.

Extension allows solutions to change, if information is preserved or increased: $(\hat{\alpha} : \star, \hat{\beta} : \star = \hat{\alpha}) \longrightarrow (\hat{\alpha} : \star = 1, \hat{\beta} : \star = \hat{\alpha})$ directly increases information about $\hat{\alpha}$, and indirectly increases information about $\hat{\beta}$. More interestingly, if $\Delta = (\hat{\alpha}{:}\star = 1, \hat{\beta}{:}\star = \hat{\alpha})$ and $\Omega = (\hat{\alpha}{:}\star = 1, \hat{\beta}{:}\star = 1)$, then $\Delta \longrightarrow \Omega$: while the solution of $\hat{\beta}$ in Ω is different, in the sense that Ω contains $\hat{\beta} : \star = 1$ while Δ contains $\hat{\beta} : \star = \hat{\alpha}$, applying Ω to the solutions gives the same result: $[\Omega]\hat{\alpha} = [\Omega]1 = 1$, which is the same as $[\Omega]1 = 1$.

Extension is quite rigid, however, in two senses. First, if a declaration appears in Γ, it appears in all extensions of Γ. Second, *extension preserves order*. For example, if $\hat{\beta}$ is declared after $\hat{\alpha}$ in Γ, then $\hat{\beta}$ will also be declared after $\hat{\alpha}$ in every extension of Γ. This holds for every variety of declaration, including equations of universal variables. This rigidity aids in enforcing type variable scoping and dependencies, which are nontrivial in a setting with higher-rank polymorphism.

5.3 Determinacy

Given appropriate inputs (Γ, e, A, p) to the algorithmic judgments, only one set of outputs (C, q, Δ) is derivable (Theorem 5 (Determinacy of Typing) in the appendix, p. 21). We use this property (for spine judgments) in the proof of soundness.

6. Soundness

We show that the algorithmic system is sound with respect to the declarative system. Soundness for the mutually-recursive judgments depends on lemmas for the auxiliary judgments (instantiation, equality elimination, checkprop, algorithmic subtyping and match coverage), which are in Appendix E.2 for space reasons.

The main soundness result has six mutually-recursive parts, one for each of the checking, synthesis, spine, and match judgments—including the principality-recovering spine judgment. We omit the parts for the match judgments; see the appendix, p. 22.

Theorem 8 (Soundness of Algorithmic Typing)**.** *Given* $\Delta \longrightarrow \Omega$*:*

(i) If $\Gamma \vdash e \Leftarrow A\ p \dashv \Delta$ *and* $\Gamma \vdash A\ p$ *type then* $[\Omega]\Delta \vdash [\Omega]e \Leftarrow [\Omega]A\ p$.

(ii) If $\Gamma \vdash e \Rightarrow A\ p \dashv \Delta$ *then* $[\Omega]\Delta \vdash [\Omega]e \Rightarrow [\Omega]A\ p$.

(iii) If $\Gamma \vdash s : A\ p \gg B\ q \dashv \Delta$ *and* $\Gamma \vdash A\ p$ *type then* $[\Omega]\Delta \vdash [\Omega]s : [\Omega]A\ p \gg [\Omega]B\ q$.

$\boxed{\Gamma \vdash e \Leftarrow A\ p \dashv \Delta}$ Under input context Γ, expression e checks against input type A, with output context Δ

$\boxed{\Gamma \vdash e \Rightarrow A\ p \dashv \Delta}$ Under input context Γ, expression e synthesizes output type A, with output context Δ

$\boxed{\begin{array}{l}\Gamma \vdash s : A\ p \gg C\ q \dashv \Delta \\ \Gamma \vdash s : A\ p \gg C\ \lceil q \rceil \dashv \Delta\end{array}}$ Under input context Γ, passing spine s to a function of type A synthesizes type C; in the $\lceil q \rceil$ form, recover principality in q if possible

$$\frac{(x : A\ p) \in \Gamma}{\Gamma \vdash x \Rightarrow [\Gamma]A\ p \dashv \Gamma}\ \mathsf{Var} \qquad \frac{\Gamma \vdash e \Rightarrow A\ q \dashv \Theta \qquad \Theta \vdash A <:^{\mathrm{pol}(B)} B \dashv \Delta}{\Gamma \vdash e \Leftarrow B\ p \dashv \Delta}\ \mathsf{Sub} \qquad \frac{\Gamma \vdash A\,!\ \mathit{type} \qquad \Gamma \vdash e \Leftarrow [\Gamma]A\,! \dashv \Delta}{\Gamma \vdash (e : A) \Rightarrow [\Delta]A\,! \dashv \Delta}\ \mathsf{Anno}$$

$$\frac{}{\Gamma \vdash () \Leftarrow 1\ p \dashv \Gamma}\ 1\mathsf{I} \qquad \frac{}{\Gamma[\hat\alpha : \star] \vdash () \Leftarrow \hat\alpha \dashv \Gamma[\hat\alpha : \star = 1]}\ 1\mathsf{I}\hat\alpha$$

$$\frac{\nu\ \mathit{chk\text{-}I} \qquad \Gamma, \alpha : \kappa \vdash \nu \Leftarrow A\ p \dashv \Delta, \alpha : \kappa, \Theta}{\Gamma \vdash \nu \Leftarrow \forall \alpha : \kappa.\ A\ p \dashv \Delta}\ \forall\mathsf{I} \qquad \frac{\Gamma, \hat\alpha : \kappa \vdash e\ s : [\hat\alpha/\alpha]A \gg C\ q \dashv \Delta}{\Gamma \vdash e\ s : \forall \alpha : \kappa.\ A\ p \gg C\ q \dashv \Delta}\ \forall\mathsf{Spine}$$

$$\frac{e \text{ not a case} \qquad \Gamma \vdash P\ \mathit{true} \dashv \Theta \qquad \Theta \vdash e \Leftarrow [\Theta]A\ p \dashv \Delta}{\Gamma \vdash e \Leftarrow A \wedge P\ p \dashv \Delta}\ \wedge\mathsf{I} \qquad \frac{\nu\ \mathit{chk\text{-}I} \qquad \Gamma, \blacktriangleright_P \,/\, P \dashv \Theta \qquad \Theta \vdash \nu \Leftarrow [\Theta]A\,! \dashv \Delta, \blacktriangleright_P, \Delta'}{\Gamma \vdash \nu \Leftarrow P \supset A\,! \dashv \Delta}\ \supset\mathsf{I}$$

$$\frac{\nu\ \mathit{chk\text{-}I} \qquad \Gamma, \blacktriangleright_P \,/\, P \dashv \bot}{\Gamma \vdash \nu \Leftarrow P \supset A\,! \dashv \Gamma}\ \supset\mathsf{I}\bot \qquad \frac{\Gamma \vdash P\ \mathit{true} \dashv \Theta \qquad \Theta \vdash e\ s : [\Theta]A\ p \gg C\ q \dashv \Delta}{\Gamma \vdash e\ s : P \supset A\ p \gg C\ q \dashv \Delta}\ \supset\mathsf{Spine} \qquad \frac{\Gamma, x : A\ p \vdash \nu \Leftarrow A\ p \dashv \Delta, x : A\ p, \Theta}{\Gamma \vdash \mathsf{rec}\ x.\nu \Leftarrow A\ p \dashv \Delta}\ \mathsf{Rec}$$

$$\frac{\Gamma, x : A\ p \vdash e \Leftarrow B\ p \dashv \Delta, x : A\ p, \Theta}{\Gamma \vdash \lambda x.\, e \Leftarrow A \to B\ p \dashv \Delta}\ \to\mathsf{I} \qquad \frac{\Gamma[\hat\alpha_1 : \star, \hat\alpha_2 : \star, \hat\alpha : \star = \hat\alpha_1 \to \hat\alpha_2], x : \hat\alpha_1 \vdash e \Leftarrow \hat\alpha_2 \dashv \Delta, x : \hat\alpha_1, \Delta'}{\Gamma[\hat\alpha : \star] \vdash \lambda x.\, e \Leftarrow \hat\alpha \dashv \Delta}\ \to\mathsf{I}\hat\alpha$$

$$\frac{\Gamma \vdash e \Rightarrow A\ p \dashv \Theta \qquad \Theta \vdash s : A\ p \gg C\ \lceil q \rceil \dashv \Delta}{\Gamma \vdash e\ s \Rightarrow C\ q \dashv \Delta}\ \to\mathsf{E} \qquad \frac{\Gamma \vdash s : A\,! \gg C \not\Downarrow \dashv \Delta \qquad \mathsf{FEV}(C) = \emptyset}{\Gamma \vdash s : A\,! \gg C\ \lceil ! \rceil \dashv \Delta}\ \mathsf{SpineRecover} \qquad \frac{\Gamma \vdash s : A\ p \gg C\ q \dashv \Delta \qquad \big((p = \not\Downarrow) \text{ or } (q = !) \text{ or } (\mathsf{FEV}(C) \neq \emptyset)\big)}{\Gamma \vdash s : A\ p \gg C\ \lceil q \rceil \dashv \Delta}\ \mathsf{SpinePass}$$

$$\frac{}{\Gamma \vdash \cdot : A\ p \gg A\ p \dashv \Gamma}\ \mathsf{EmptySpine} \qquad \frac{\Gamma \vdash e \Leftarrow A\ p \dashv \Theta \qquad \Theta \vdash s : [\Theta]B\ p \gg C\ q \dashv \Delta}{\Gamma \vdash e\ s : A \to B\ p \gg C\ q \dashv \Delta}\ \to\mathsf{Spine}$$

$$\frac{\Gamma[\hat\alpha_2 : \star, \hat\alpha_1 : \star, \hat\alpha : \star = \hat\alpha_1 \to \hat\alpha_2] \vdash e\ s : (\hat\alpha_1 \to \hat\alpha_2) \gg C \dashv \Delta}{\Gamma[\hat\alpha : \star] \vdash e\ s : \hat\alpha \gg C \dashv \Delta}\ \hat\alpha\mathsf{Spine} \qquad \frac{\Gamma \vdash e \Rightarrow A\,! \dashv \Theta \qquad \Theta \vdash \Pi :: [\Theta]A \Leftarrow [\Theta]C\ p \dashv \Delta \qquad \Delta \vdash \Pi\ \mathit{covers}\ [\Delta]A}{\Gamma \vdash \mathsf{case}(e, \Pi) \Leftarrow C\ p \dashv \Delta}\ \mathsf{Case}$$

Figure 11. Algorithmic typing, omitting rules for $\times$, $+$, and Vec

(iv) If $\Gamma \vdash s : A\ p \gg B\ \lceil q \rceil \dashv \Delta$ *and* $\Gamma \vdash A\ p$ *type then* $[\Omega]\Delta \vdash [\Omega]s : [\Omega]A\ p \gg [\Omega]B\ \lceil q \rceil$.

Much of this proof is simply "turning the crank": applying the induction hypothesis to each premise, yielding derivations of corresponding declarative judgments (with Ω applied to everything in sight), and applying the corresponding declarative rule; for example, in the Sub case we finish the proof by applying DeclSub. The SpineRecover case is interesting: we do finish by applying DeclSpineRecover, but since DeclSpineRecover contains a premise that quantifies over all declarative derivations of a certain form, we must appeal to completeness! Consequently, soundness and completeness are really two parts of one theorem.

These parts are mutually recursive—later, we'll see that the DeclSpineRecover case of completeness must appeal to soundness (to show that the algorithmic type has no free existential variables). We cannot induct on the given derivation alone, because the derivations in the "for all" part of DeclSpineRecover are not subderivations. So we need a more involved induction measure that can make the leaps between soundness and completeness: lexicographic order with (1) the size of the subject term, (2) the judgment form, with ordinary spine judgments considered smaller than recovering spine judgments, and (3) the height of the derivation:

$$\left\langle e/s/\Pi,\ \begin{array}{c}\text{ordinary spine judgment} \\ < \\ \text{recovering spine judgment}\end{array},\ \mathrm{height}(\mathcal{D}) \right\rangle$$

Proof sketch*—SpineRecover *case. By i.h., $[\Omega]\Gamma \vdash [\Omega]s : [\Omega]A\,! \gg [\Omega]C\ q$. Our goal is to apply DeclSpineRecover, which requires that we show that for *all* C' such that $[\Omega]\Theta \vdash s : [\Omega]A\,! \gg C' \not\Downarrow$, we have $C' = [\Omega]C$. Suppose we have such a C'. By completeness (Theorem 11), $\Gamma \vdash s : [\Gamma]A\,! \gg C''\ q \dashv \Delta''$ where $\Delta'' \longrightarrow \Omega''$. We already have (as a subderivation) $\Gamma \vdash s : A\,! \gg C \not\Downarrow \dashv \Delta$, so by determinacy, $C'' = C$ and $q = \not\Downarrow$ and $\Delta'' = \Delta$. With the help of lemmas about context application, we can show $C' = [\Omega'']C'' = [\Omega'']C = [\Omega]C$. (Using completeness is permitted since our measure says a non-principality-restoring judgment is smaller.)

7. Completeness

We show that the algorithmic system is complete with respect to the declarative system. As with soundness, we need to show completeness of the auxiliary algorithmic judgments. We omit the full statements of these lemmas; as an example, if $[\Omega]\hat\alpha = [\Omega]\tau$ and $\hat\alpha \notin \mathrm{FV}(\tau)$ then $\Gamma \vdash \hat\alpha := \tau : \kappa \dashv \Delta$.

7.1 Separation

To show completeness, we will need to show that wherever the declarative rule DeclSpineRecover is applied, we can apply the algorithmic rule SpineRecover. Thus, we need to show that *semantic* principality—that no other type can be given—entails that a type has no free existential variables.

The principality-recovering rules are potentially applicable when we start with a principal type A ! but produce C $\not!$, with Decl$\forall$Spine changing ! to $\not!$. Completeness (Thm. 11) will use the "for all" part of DeclSpineRecover, which quantifies over all types produced by the spine rules under a given declarative context $[\Omega]\Gamma$. By i.h. we get an algorithmic spine judgment $\Gamma \vdash s : A' \,!\gg C' \not! \dashv \Delta$. Since A' is principal, unsolved existentials in C' must have been introduced within this derivation—they can't be in Γ already. Thus, we might have $\hat{\alpha} : \star \vdash s : A' \,!\gg \hat{\beta} \not! \dashv \hat{\alpha} : \star, \hat{\beta} : \star$ where a Decl$\forall$Spine subderivation introduced $\hat{\beta}$, but $\hat{\alpha}$ can't appear in C'. We also can't equate $\hat{\alpha}$ and $\hat{\beta}$ in Δ, which would be tantamount to $C' = \hat{\alpha}$. Knowing that unsolved existentials in C' are "new" and independent from those in Γ means we can argue that, if there *were* an unsolved existential in C', it would correspond to an unforced choice in a Decl$\forall$Spine subderivation, invalidating the "for all" part of DeclSpineRecover. Formalizing claims like "must have been introduced" requires several definitions.

Definition 1 (Separation)**.**
An algorithmic context Γ *is* separable into $\Gamma_L * \Gamma_R$ *if (1)* $\Gamma = (\Gamma_L, \Gamma_R)$ *and (2) for all* $(\hat{\alpha} : \kappa = \tau) \in \Gamma_R$ *it is the case that* $\mathsf{FEV}(\tau) \subseteq \mathsf{dom}(\Gamma_R)$.

If Γ is separable into $\Gamma_L * \Gamma_R$, then Γ_R is self-contained in the sense that all existential variables declared in Γ_R have solutions whose existential variables are themselves declared in Γ_R. Every context Γ is separable into $\cdot * \Gamma$ and into $\Gamma * \cdot$.

Definition 2 (Separation-Preserving Extension)**.**
The separated context $\Gamma_L * \Gamma_R$ *extends to* $\Delta_L * \Gamma_R$*, written* $(\Gamma_L * \Gamma_R) \xrightarrow[*]{} (\Delta_L * \Delta_R)$*, if* $(\Gamma_L, \Gamma_R) \longrightarrow (\Delta_L, \Delta_R)$ *and* $\mathsf{dom}(\Gamma_L) \subseteq \mathsf{dom}(\Delta_L)$ *and* $\mathsf{dom}(\Gamma_R) \subseteq \mathsf{dom}(\Delta_R)$.

Separation-preserving extension says that variables from one side of $*$ haven't "jumped" to the other side. Thus, Δ_L may add existential variables to Γ_L, and Δ_R may add existential variables to Γ_R, but no variable from Γ_L ends up in Δ_R and no variable from Γ_R ends up in Δ_L. It is necessary to write $(\Gamma_L * \Gamma_R) \xrightarrow[*]{} (\Delta_L * \Delta_R)$ rather than $(\Gamma_L * \Gamma_R) \longrightarrow (\Delta_L * \Delta_R)$, because only $\xrightarrow[*]{}$ includes the domain conditions. For example, $(\hat{\alpha} * \hat{\beta}) \longrightarrow (\hat{\alpha}, \hat{\beta} = \hat{\alpha}) * \cdot$, but $\hat{\beta}$ has jumped to the left of $*$ in the context $(\hat{\alpha}, \hat{\beta} = \hat{\alpha}) * \cdot$.

We prove many lemmas about separation, but use only one of them in the subsequent development (in the DeclSpineRecover case of typing completeness), and then only the part for spines. It says that if we have a spine whose type A mentions only variables in Γ_R, then the output context Δ extends Γ and preserves separation, and the output type C mentions only variables in Δ_R:

Lemma (Separation—Main)**.**
If $\Gamma_L * \Gamma_R \vdash s : A\ p \gg C\ q \dashv \Delta$ *or* $\Gamma_L * \Gamma_R \vdash s : A\ p \gg C\ \lceil q \rceil \dashv \Delta$ *and* $\Gamma_L * \Gamma_R \vdash A\ p$ *type and* $\mathsf{FEV}(A) \subseteq \mathsf{dom}(\Gamma_R)$ *then* $\Delta = (\Delta_L * \Delta_R)$ *and* $(\Gamma_L * \Gamma_R) \xrightarrow[*]{} (\Delta_L * \Delta_R)$ *and* $\mathsf{FEV}(C) \subseteq \mathsf{dom}(\Delta_R)$.

7.2 Completeness of typing

Like soundness, completeness has six mutually-recursive parts. Again, the match judgments are in the appendix, p. 22.

Theorem 11 (Completeness of Algorithmic Typing)**.**
Given $\Gamma \longrightarrow \Omega$ *such that* $\mathsf{dom}(\Gamma) = \mathsf{dom}(\Omega)$*:*

(i) If $\Gamma \vdash A\ p$ *type and* $[\Omega]\Gamma \vdash [\Omega]e \Leftarrow [\Omega]A\ p$ *and* $p' \sqsubseteq p$ *then there exist* Δ *and* Ω' *such that* $\Delta \longrightarrow \Omega'$ *and* $\mathsf{dom}(\Delta) = \mathsf{dom}(\Omega')$ *and* $\Omega \longrightarrow \Omega'$ *and* $\Gamma \vdash e \Leftarrow [\Gamma]A\ p' \dashv \Delta$.

(ii) If $\Gamma \vdash A\ p$ *type and* $[\Omega]\Gamma \vdash [\Omega]e \Rightarrow A\ p$ *then there exist* Δ, Ω', A', *and* $p' \sqsubseteq p$ *such that* $\Delta \longrightarrow \Omega'$ *and* $\mathsf{dom}(\Delta) = \mathsf{dom}(\Omega')$ *and* $\Omega \longrightarrow \Omega'$ *and* $\Gamma \vdash e \Rightarrow A'\ p' \dashv \Delta$ *and* $A' = [\Delta]A'$ *and* $A = [\Omega']A'$.

(iii) If $\Gamma \vdash A\ p$ *type and* $[\Omega]\Gamma \vdash [\Omega]s : [\Omega]A\ p \gg B\ q$ *and* $p' \sqsubseteq p$ *then there exist* Δ, Ω', B', *and* $q' \sqsubseteq q$ *such that* $\Delta \longrightarrow \Omega'$ *and* $\mathsf{dom}(\Delta) = \mathsf{dom}(\Omega')$ *and* $\Omega \longrightarrow \Omega'$ *and* $\Gamma \vdash s : [\Gamma]A\ p' \gg B'\ q' \dashv \Delta$ *and* $B' = [\Delta]B'$ *and* $B = [\Omega']B'$.

(iv) As part (iii), but with $\gg B\ \lceil q \rceil \cdots$ *and* $\gg B'\ \lceil q' \rceil \cdots$.

Proof sketch*—DeclSpineRecover *case. By i.h., $\Gamma \vdash s : [\Gamma]A \,!\gg C' \not! \dashv \Delta$ where $\Delta \longrightarrow \Omega'$ and $\Omega \longrightarrow \Omega'$ and $\mathsf{dom}(\Delta) = \mathsf{dom}(\Omega')$ and $C = [\Omega']C'$.

To apply SpineRecover, we need to show $\mathsf{FEV}([\Delta]C') = \emptyset$. Suppose, for a contradiction, that $\mathsf{FEV}([\Delta]C') \neq \emptyset$. Construct a variant of Ω' called Ω_2 that has a different solution for some $\hat{\alpha} \in \mathsf{FEV}([\Delta]C')$. By soundness (Thm. 11), $[\Omega_2]\Gamma \vdash [\Omega_2]s : [\Omega_2]A \,!\gg [\Omega_2]C' \not!$. Using a separation lemma with the trivial separation $\Gamma = (\Gamma * \cdot)$ we get $\Delta = (\Delta_L * \Delta_R)$ and $(\Gamma * \cdot) \xrightarrow[*]{} (\Delta_L * \Delta_R)$ and $\mathsf{FEV}(C') \subseteq \mathsf{dom}(\Delta_R)$. That is, all existentials in C' were introduced within the derivation of the (algorithmic) spine judgment. Thus, applying Ω_2 to things gives the same result as Ω, except for C', giving $[\Omega]\Gamma \vdash [\Omega]s : [\Omega]A \,!\gg [\Omega_2]C' \not!$. Now instantiate the "for all C_2" premise with $C_2 = [\Omega_2]C'$, giving $C = [\Omega_2]C'$. But we chose Ω_2 to have a different solution for $\hat{\alpha} \in \mathsf{FEV}(C')$, so we have $C \neq [\Omega_2]C'$: Contradiction. Therefore $\mathsf{FEV}([\Delta]C') = \emptyset$, so we can apply SpineRecover.

8. Discussion and Related Work

A staggering amount of work has been done on GADTs and indexed types, and for space reasons we cannot offer a comprehensive survey of the literature. So we compare more deeply to fewer papers, to communicate our understanding of the design space.

Proof theory and type theory. As described in Section 1, there are two logical accounts of equality—the identity type of Martin-Löf and the equality type of Schroeder-Heister (1994) and Girard (1992). The Girard/Schroeder-Heister equality has a more direct connection to pattern matching, which is why we make use of it. Coquand (1996) pioneered the study of pattern matching in dependent type theory. One perhaps surprising feature of Coquand's pattern-matching syntax is that it is strictly stronger than Martin-Löf's eliminators. His rules can derive the uniqueness of identity proofs as well as the disjointness of constructors. Constructor disjointness is also derivable from the Girard/Schroeder-Heister equality, because there is no unifier for two distinct constructors.

In future work, we hope to study the relation between these two notions of equality in more depth; richer equational theories (such as the theory of commutative rings or the $\beta\eta$-theory of the lambda calculus) do not have decidable unification, but it seems plausible that there are hybrid approaches which might let us retain some of the convenience of the G/SH equality rule while retaining the decidability of Martin-Löf's J eliminator.

Indexed and refinement types. Dependent ML (Xi and Pfenning 1999) indexed programs with propositional constraints, extending the ML type discipline to maintain additional invariants tracked by the constraints. DML collected constraints from the program and passed them to a constraint solver, a technique used by systems like Stardust (Dunfield 2007) and liquid types (Rondon et al. 2008).

From phantom types to GADTs. Leijen and Meijer (1999) introduced the term *phantom type* to describe a technique for programming in ML/Haskell where additional type parameters are used to constrain when values are well-typed. This idea proved to have many applications, ranging from foreign function interfaces (Blume 2001) to encoding Java-style subtyping (Fluet and Pucella 2006). Phantom types allow *constructing* values with constrained types, but do not easily permit *learning* about type equalities by *analyzing* them, putting applications such as intensional type analysis (Harper and Morrisett 1995) out of reach. Both Cheney and

Hinze (2003) and Xi et al. (2003) proposed treating equalities as a first-class concept, giving explicitly-typed calculi for equalities, but without studying algorithms for type inference.

Simonet and Pottier (2007) gave a constraint-based algorithm for type inference for GADTs. It is this work which first identified the potential intractibility of type inference arising from the interaction of hypothetical constraints and unification variables. To resolve this issue they introduce the notion of *tractable* constraints (i.e., constraints where hypothetical equations never contain existentials), and require placing enough annotations that all constraints are tractable. In general, this could require annotations on case expressions, so subsequent work focused on relaxing this requirement. Though quite different in technical detail, stratified inference (Pottier and Régis-Gianas 2006) and *wobbly types* (Peyton Jones et al. 2006) both work by pushing type information from annotations to case expressions, with stratified type inference literally moving annotations around, and wobbly types tracking which parts of a type have no unification variables. Modern GHC uses the OutsideIn algorithm (Vytiniotis et al. 2011), which further relaxes the constraint: as long as case analysis cannot modify what is known about an equation, the case analysis is permitted.

In our type system, the checking judgment of the bidirectional algorithm serves to propagate annotations, and our requirement that the scrutinee of a case expression be principal ensures that no equations contain unification variables. This is close in effect to stratified types, and is less expressive than OutsideIn. This is a deliberate design choice to keep the declarative specification simple, rather than an inherent limit of our approach. To specify the OutsideIn approach, the case rule in our declarative system should permit scrutinizing an expression if all types that can be synthesized for it have exactly the same equations, even if they differ in their monotype parts. We thought such a spec is harder for programmers to develop an intuition for than simply saying that a scrutinee must synthesize a unique type. However, the technique we use—higher-order rules with implicational premises like DeclSpineRecover—should work for this case.

More recently, Garrigue and Rémy (2013) proposed *ambivalent types*, which are a way of deciding when it is safe to generalize the type of a function using GADTs. This idea is orthogonal to our calculus, simply because we do no generalization at all: *every* polymorphic function takes an annotation. However, Garrigue and Rémy (2013) also emphasize the importance of *monotonicity*, which says that substitution should be stable under subtyping, that is, giving a more general type should not cause subtyping to fail. This condition is satisfied by our bidirectional system.

Karachalias et al. (2015) developed a coverage algorithm for GADTs that depends on external constraint solving; we offer a more self-contained but still logically-motivated approach.

Extensions. To keep our formalization manageable, we left out some features that would be desirable in practice. In particular, we need (1) type constructors which take arguments and (2) recursive types (Pierce 2002, chapter 20). The primary issue with both of these features is that they need to permit using existentials and other "large" type connectives, and our system seemingly relies on monotypes (which cannot contain such connectives). This limitation should create no difficulties in typical practice, if we treat user-defined type constructors, such as `List`, as monotypes and expand the definition only as needed: when checking an expression against a user type constructor, and when demanded by pattern matching.

References

Andreas Abel, Thierry Coquand, and Peter Dybjer. Verifying a semantic $\beta\eta$-conversion test for Martin-Löf type theory. In *Mathematics of Program Construction (MPC'08)*, volume 5133 of *LNCS*, pages 29–56, 2008.

Matthias Blume. No-longer-foreign: Teaching an ML compiler to speak C "natively". *Electronic Notes in Theoretical Computer Science*, 59(1), 2001.

James Cheney and Ralf Hinze. First-class phantom types. Technical Report CUCIS TR2003-1901, Cornell University, 2003.

Thierry Coquand. An algorithm for type-checking dependent types. *Science of Computer Programming*, 26(1–3):167–177, 1996.

Rowan Davies and Frank Pfenning. Intersection types and computational effects. In *ICFP*, pages 198–208, 2000.

Joshua Dunfield. Refined typechecking with Stardust. In *Programming Languages meets Programming Verification (PLPV '07)*, 2007.

Joshua Dunfield and Neelakantan R. Krishnaswami. Complete and easy bidirectional typechecking for higher-rank polymorphism. In *ICFP*, 2013. arXiv:`1306.6032 [cs.PL]`.

Joshua Dunfield and Frank Pfenning. Type assignment for intersections and unions in call-by-value languages. In *FoSSaCS*, pages 250–266, 2003.

Matthew Fluet and Riccardo Pucella. Phantom types and subtyping. arXiv:`cs/0403034 [cs.PL]`, 2006.

Jacques Garrigue and Didier Rémy. Ambivalent types for principal type inference with GADTs. In *APLAS*, 2013.

Jean-Yves Girard. A fixpoint theorem in linear logic. Post to Linear Logic mailing list, `http://www.seas.upenn.edu/~sweirich/types/archive/1992/msg00030.html`, 1992.

Robert Harper and Greg Morrisett. Compiling polymorphism using intensional type analysis. In *POPL*, pages 130–141. ACM Press, 1995.

Georgios Karachalias, Tom Schrijvers, Dimitrios Vytiniotis, and Simon Peyton Jones. GADTs meet their match: pattern-matching warnings that account for GADTs, guards, and laziness. In *ICFP*, pages 424–436, 2015.

Neelakantan R. Krishnaswami. Focusing on pattern matching. In *POPL*, pages 366–378. ACM Press, 2009.

Konstantin Läufer and Martin Odersky. Polymorphic type inference and abstract data types. *ACM Trans. Prog. Lang. Sys.*, 16(5):1411–1430, 1994.

Daan Leijen and Erik Meijer. Domain specific embedded compilers. In *USENIX Conf. Domain-Specific Languages (DSL '99)*, pages 109–122, 1999.

Dale Miller. Unification under a mixed prefix. *J. Symbolic Computation*, 14(4): 321–358, 1992.

Martin Odersky, Matthias Zenger, and Christoph Zenger. Colored local type inference. In *POPL*, pages 41–53, 2001.

Simon Peyton Jones, Dimitrios Vytiniotis, Stephanie Weirich, and Geoffrey Washburn. Simple unification-based type inference for GADTs. In *ICFP*, pages 50–61, 2006.

Simon Peyton Jones, Dimitrios Vytiniotis, Stephanie Weirich, and Mark Shields. Practical type inference for arbitrary-rank types. *J. Functional Programming*, 17(1):1–82, 2007.

Brigitte Pientka. A type-theoretic foundation for programming with higher-order abstract syntax and first-class substitutions. In *POPL*, pages 371–382, 2008.

Benjamin C. Pierce. *Types and Programming Languages*. MIT Press, 2002.

Benjamin C. Pierce and David N. Turner. Local type inference. *ACM Trans. Prog. Lang. Sys.*, 22:1–44, 2000.

François Pottier and Yann Régis-Gianas. Stratified type inference for generalized algebraic data types. In *POPL*, pages 232–244, 2006.

Patrick Rondon, Ming Kawaguchi, and Ranjit Jhala. Liquid types. In *PLDI*, pages 159–169, 2008.

Peter Schroeder-Heister. Definitional reflection and the completion. In *Extensions of Logic Programming*, LNCS, pages 333–347. Springer, 1994.

Vincent Simonet and François Pottier. A constraint-based approach to guarded algebraic data types. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 29(1):1, 2007.

Dimitrios Vytiniotis, Simon Peyton Jones, and Tom Schrijvers. Let should not be generalised. In *Workshop on Types in Language Design and Impl. (TLDI '10)*, pages 39–50, 2010.

Dimitrios Vytiniotis, Simon Peyton Jones, Tom Schrijvers, and Martin Sulzmann. OutsideIn(X): Modular type inference with local assumptions. *J. Functional Programming*, 21(4–5):333–412, 2011.

Kevin Watkins, Iliano Cervesato, Frank Pfenning, and David Walker. A concurrent logical framework: The propositional fragment. In *Types for Proofs and Programs*, pages 355–377. Springer LNCS 3085, 2004.

Hongwei Xi and Frank Pfenning. Dependent types in practical programming. In *POPL*, pages 214–227, 1999.

Hongwei Xi, Chiyan Chen, and Gang Chen. Guarded recursive datatype constructors. In *POPL*, pages 224–235, 2003.

Appendix: LICS submission 39

This file contains rules, figures and definitions omitted in the main paper for space reasons, as well as statements of theorems and a few selected lemmas. Statements of all lemmas, and complete proofs, are in another, much longer, file:

github.com/joshuadunfield/lics39/raw/master/lics39_proofs.pdf

A. Additional Examples

A.1 Spine recovery

Suppose we have an identity function id, defined in an algorithmic context Γ by the hypothesis $\mathtt{id} : (\forall\alpha : \star.\ \alpha \to \alpha)\ !$. Since the hypothesis has !, the type of id is known to be principal. If we apply id to the unit (), we expect to get something of unit type 1. Despite the $\forall$ quantifier in the type of id, the resulting type should be principal, because no other type is possible. We can indeed derive that type in our system:

$$\dfrac{\dfrac{(\mathtt{id} : (\forall\alpha : \star.\ \alpha \to \alpha)\ !) \in \Gamma}{\Gamma \vdash \mathtt{id} \Rightarrow (\forall\alpha : \star.\ \alpha \to \alpha)\ ! \dashv \Gamma}\ \mathsf{Var} \qquad \Gamma \vdash (\,)\cdot) : (\forall\alpha : \star.\ \alpha \to \alpha)\ ! \gg 1\ \lceil !\rceil \dashv \Gamma, \hat\alpha : \star = 1}{\Gamma \vdash \mathtt{id}\ (\,)\cdot) \Rightarrow 1\ ! \dashv \Gamma, \hat\alpha : \star = 1}\ \to\mathsf{E}$$

(Here, we write the application id () as id (() ·), to show the structure of the spine as analyzed by the typing rules.)

In the derivation of the second premise of $\to$E, shown below, we can follow the evolution of the principality marker.

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma, \hat\alpha : \star \vdash ()\ \Leftarrow \hat\alpha\ \not! \dashv \Gamma, \hat\alpha : \star = 1}\ 1\mathsf{I}\hat\alpha \qquad \dfrac{}{\Gamma, \hat\alpha : \star = 1 \vdash \cdot : 1\ \not! \gg 1\ \not! \dashv \Gamma, \hat\alpha : \star = 1}\ \mathsf{EmptySpine}}{\Gamma, \hat\alpha : \star \vdash (()\cdot) : \hat\alpha \to \hat\alpha\ \not! \gg 1\ \not! \dashv \Gamma, \hat\alpha : \star = 1}\ \to\mathsf{Spine}}{\Gamma \vdash (()\cdot) : (\forall\alpha : \star.\ \alpha \to \alpha)\ ! \gg 1\ \not! \dashv \Gamma, \hat\alpha : \star = 1}\ \forall\mathsf{Spine} \qquad \mathsf{FEV}(1) = \emptyset}{\Gamma \vdash (()\cdot) : (\forall\alpha : \star.\ \alpha \to \alpha)\ \underbrace{!}_{\text{input}} \gg 1\ \lceil !\rceil \dashv \Gamma, \hat\alpha : \star = 1}\ \mathsf{SpineRecover}$$

- The input principality (marked "input") is !, because the input type $(\forall\alpha : \star.\ \alpha \to \alpha)$ was marked as principal in the hypothesis typing id.
- Rule SpineRecover begins by invoking the ordinary (non-recovering) spine judgment, passing all inputs unchanged, including the principality !.
- Rule $\forall$Spine adds an existential variable $\hat\alpha$ to represent the instantiation of the quantified type variable α, and substitutes $\hat\alpha$ for α. Since this instantiation is, in general, *not* principal, it replaces ! with $\not!$ (highlighted) in its premise. This marks the type $\hat\alpha \to \hat\alpha$ as non-principal.
- Rule $\to$Spine decomposes $\hat\alpha \to \hat\alpha$ and checks () against $\hat\alpha$, maintaining the principality $\not!$. Once principality is lost, it can only be recovered within the SpineRecover rule itself.
- Rule $1\mathsf{I}\hat\alpha$ notices that we are checking () against an unknown type $\hat\alpha$; since the expression is (), the type $\hat\alpha$ must be 1, so it adds that solution to its output context.
- Moving to the second premise of $\to$Spine, we analyze the remaining part of the spine. That is just the empty spine ·, and rule EmptySpine passes its inputs along as outputs. In particular, the principality $\not!$ is unchanged.
- The principalities are passed down the derivation to the conclusion of $\forall$Spine, where $\not!$ is highlighted.
- In SpineRecover, we notice that the output type 1 has no existential variables ($\mathsf{FEV}(1) = \emptyset$), which allows us to recover principality of the output type: $\lceil !\rceil$.

In the corresponding derivation in our declarative system, we have, instead, a check that no other types are derivable:

$$\dfrac{\Psi \vdash 1 : \star \qquad \dfrac{\dfrac{\dfrac{}{\Psi \vdash ()\ \Leftarrow 1\ \not!}\ \mathsf{Decl1I} \qquad \dfrac{}{\Psi \vdash \cdot : 1\ \not! \gg 1\ \not!}\ \mathsf{DeclEmptySpine}}{\Psi \vdash (()\cdot) : 1 \to 1\ \not! \gg 1\ \not!}\ \mathsf{Decl}\to\mathsf{Spine}}{\Psi \vdash (()\cdot) : (\forall\alpha : \star.\ \alpha \to \alpha)\ ! \gg 1\ \not!}\ \mathsf{Decl}\forall\mathsf{Spine} \qquad \begin{array}{l}\text{for all } C'.\\ \text{if } \Psi \vdash (()\cdot) : (\forall\alpha : \star.\ \alpha \to \alpha)\ ! \gg C'\ \not!\\ \text{then } C' = 1\end{array}}{\Psi \vdash (()\cdot) : (\forall\alpha : \star.\ \alpha \to \alpha)\ \underbrace{!}_{\text{input}} \gg 1\ \lceil !\rceil}\ \mathsf{DeclSpineRecover}$$

Here, we highlight the replacement in Decl$\forall$Spine of the quantified type variable α by the "guessed" solution 1. The second premise of DeclSpineRecover checks that no other output type C' could have been produced, no matter what solution was chosen by Decl$\forall$Spine for α.

B. Additional Notation

Our proofs use some additional notation not described in the main paper:

Two-hole contexts. Occasionally, we also need contexts with *two* ordered holes:

$$\Gamma = \Gamma_0[\Theta_1][\Theta_2] \quad \text{means } \Gamma \text{ has the form } (\Gamma_L, \Theta_1, \Gamma_M, \Theta_2, \Gamma_R)$$

C. Context Extension

The rules deriving the context extension judgment (Figure 14) say that the empty context extends the empty context (⟶Id); a term variable typing with A' extends one with A if applying the extending context Δ to A and A' yields the same type (⟶Var); universal variable declarations and equations must match (⟶Uvar, ⟶Eqn); scope markers must match (⟶Marker); and, existential variables may either match (⟶Unsolved, ⟶Solved), get solved by the extending context (⟶Solve), or be added by the extending context (⟶Add, ⟶AddSolved).

D. Figures

We repeat some figures from the main paper. In Figures 6a and 11a, we include rules omitted from the main paper for space reasons.

$\boxed{\Psi \vdash e \Leftarrow A\ p}$ Under context Ψ, expression e checks against input type A

$\boxed{\Psi \vdash e \Rightarrow A\ p}$ Under context Ψ, expression e synthesizes output type A

$\boxed{\Psi \vdash s : A\ p \gg C\ q}$ $\boxed{\Psi \vdash s : A\ p \gg C\ \lceil q \rceil}$ Under context Ψ, passing spine s to a function of type A synthesizes type C; in the $\lceil q \rceil$ form, recover principality in q if possible

$\boxed{\Psi \vdash P\ \mathit{true}}$ Under context Ψ, check P

$$\frac{}{\Psi \vdash (t = t)\ \mathit{true}}\ \mathsf{DeclCheckpropEq}$$

$$\frac{x : A\ p \in \Psi}{\Psi \vdash x \Rightarrow A\ p}\ \mathsf{DeclVar} \qquad \frac{\Psi \vdash e \Rightarrow A\ q \qquad \Psi \vdash A \leq^{\mathrm{pol}(B)} B}{\Psi \vdash e \Leftarrow B\ p}\ \mathsf{DeclSub} \qquad \frac{\Psi \vdash A\ \mathit{type} \qquad \Psi \vdash e \Leftarrow A\ !}{\Psi \vdash (e : A) \Rightarrow A\ !}\ \mathsf{DeclAnno}$$

$$\frac{}{\Psi \vdash () \Leftarrow 1\ p}\ \mathsf{Decl1I} \qquad \frac{v\ \mathit{chk\text{-}I} \qquad \Psi, \alpha : \kappa \vdash v \Leftarrow A\ p}{\Psi \vdash v \Leftarrow (\forall \alpha : \kappa.\ A)\ p}\ \mathsf{Decl}\forall\mathsf{I} \qquad \frac{\Psi \vdash \tau : \kappa \qquad \Psi \vdash e\ s : [\tau/\alpha]A \not\gg C\ q}{\Psi \vdash e\ s : (\forall \alpha : \kappa.\ A)\ p \gg C\ q}\ \mathsf{Decl}\forall\mathsf{Spine}$$

$$\frac{\Psi \vdash P\ \mathit{true} \qquad \Psi \vdash e \Leftarrow A\ p}{\Psi \vdash e \Leftarrow (A \wedge P)\ p}\ \mathsf{Decl}\wedge\mathsf{I} \qquad \frac{v\ \mathit{chk\text{-}I} \qquad \Psi \,/\, P \vdash v \Leftarrow A\ !}{\Psi \vdash v \Leftarrow (P \supset A)\ !}\ \mathsf{Decl}\supset\mathsf{I} \qquad \frac{\Psi \vdash P\ \mathit{true} \qquad \Psi \vdash e\ s : A\ p \gg C\ q}{\Psi \vdash e\ s : (P \supset A)\ p \gg C\ q}\ \mathsf{Decl}\supset\mathsf{Spine}$$

$$\frac{\Psi, x : A\ p \vdash v \Leftarrow A\ p}{\Psi \vdash \mathsf{rec}\ x.\ v \Leftarrow A\ p}\ \mathsf{DeclRec} \qquad \frac{\Psi, x : A\ p \vdash e \Leftarrow B\ p}{\Psi \vdash \lambda x.\ e \Leftarrow A \to B\ p}\ \mathsf{Decl}\to\mathsf{I} \qquad \frac{\Psi \vdash e \Rightarrow A\ p \qquad \Psi \vdash s : A\ p \gg C\ \lceil q \rceil}{\Psi \vdash e\ s \Rightarrow C\ q}\ \mathsf{Decl}\to\mathsf{E}$$

$$\frac{\Psi \vdash s : A\ ! \gg C \not\!\!\downarrow \qquad \text{for all } C'.\ \text{if } \Psi \vdash s : A\ ! \gg C' \not\!\!\downarrow \text{ then } C' = C}{\Psi \vdash s : A\ ! \gg C\ \lceil ! \rceil}\ \mathsf{DeclSpineRecover} \qquad \frac{\Psi \vdash s : A\ p \gg C\ q}{\Psi \vdash s : A\ p \gg C\ \lceil q \rceil}\ \mathsf{DeclSpinePass}$$

$$\frac{}{\Psi \vdash \cdot : A\ p \gg A\ p}\ \mathsf{DeclEmptySpine} \qquad \frac{\Psi \vdash e \Leftarrow A\ p \qquad \Psi \vdash s : B\ p \gg C\ q}{\Psi \vdash e\ s : A \to B\ p \gg C\ q}\ \mathsf{Decl}\to\mathsf{Spine}$$

$$\frac{\Psi \vdash e \Leftarrow A_k\ p}{\Psi \vdash \mathsf{inj}_k\ e \Leftarrow A_1 + A_2\ p}\ \mathsf{Decl}+\mathsf{I}_k \qquad \frac{\Psi \vdash e_1 \Leftarrow A_1\ p \qquad \Psi \vdash e_2 \Leftarrow A_2\ p}{\Psi \vdash \langle e_1, e_2 \rangle \Leftarrow A_1 \times A_2\ p}\ \mathsf{Decl}\times\mathsf{I}$$

$$\frac{\Psi \vdash t = \mathsf{zero}\ \mathit{true}}{\Psi \vdash [] \Leftarrow (\mathsf{Vec}\ t\ A)\ p}\ \mathsf{DeclNil} \qquad \frac{\Psi \vdash t = \mathsf{succ}(t_2)\ \mathit{true} \qquad \Psi \vdash e_1 \Leftarrow A\ p \qquad \Psi \vdash e_2 \Leftarrow (\mathsf{Vec}\ t_2\ A) \not\!\!\downarrow}{\Psi \vdash e_1 :: e_2 \Leftarrow (\mathsf{Vec}\ t\ A)\ p}\ \mathsf{DeclCons}$$

$$\frac{\Psi \vdash e \Rightarrow A\ ! \qquad \Psi \vdash \Pi :: A \Leftarrow C\ p \qquad \Psi \vdash \Pi\ \mathit{covers}\ A}{\Psi \vdash \mathsf{case}(e, \Pi) \Leftarrow C\ p}\ \mathsf{DeclCase}$$

$\boxed{\Psi \,/\, P \vdash e \Leftarrow C\ p}$ Under context Ψ, incorporate proposition P and check e against C

$$\frac{\mathsf{mgu}(\sigma, \tau) = \bot}{\Psi \,/\, (\sigma = \tau) \vdash e \Leftarrow C\ p}\ \mathsf{DeclCheck}\bot \qquad \frac{\mathsf{mgu}(\sigma, \tau) = \theta \qquad \theta(\Psi) \vdash \theta(e) \Leftarrow \theta(C)\ p}{\Psi \,/\, (\sigma = \tau) \vdash e \Leftarrow C\ p}\ \mathsf{DeclCheckUnify}$$

Figure 6a. Declarative typing, including rules omitted from main paper

$\boxed{\Gamma \vdash e \Leftarrow A\ p \dashv \Delta}$ Under input context Γ, expression e checks against input type A, with output context Δ

$\boxed{\Gamma \vdash e \Rightarrow A\ p \dashv \Delta}$ Under input context Γ, expression e synthesizes output type A, with output context Δ

$\boxed{\Gamma \vdash s : A\ p \gg C\ q \dashv \Delta}$
$\boxed{\Gamma \vdash s : A\ p \gg C\ \lceil q \rceil \dashv \Delta}$
Under input context Γ,
passing spine s to a function of type A synthesizes type C;
in the $\lceil q \rceil$ form, recover principality in q if possible

$$\frac{(x : A\,p) \in \Gamma}{\Gamma \vdash x \Rightarrow [\Gamma]A\ p \dashv \Gamma}\ \mathsf{Var} \qquad \frac{\Gamma \vdash e \Rightarrow A\ q \dashv \Theta \qquad \Theta \vdash A <:^{\mathrm{pol}(B)} B \dashv \Delta}{\Gamma \vdash e \Leftarrow B\ p \dashv \Delta}\ \mathsf{Sub} \qquad \frac{\Gamma \vdash A\,!\ \mathit{type} \qquad \Gamma \vdash e \Leftarrow [\Gamma]A\ !\ \dashv \Delta}{\Gamma \vdash (e : A) \Rightarrow [\Delta]A\ !\ \dashv \Delta}\ \mathsf{Anno}$$

$$\frac{}{\Gamma \vdash ()\ \Leftarrow 1\ p \dashv \Gamma}\ 1\mathsf{I} \qquad \frac{}{\Gamma[\hat\alpha : \star] \vdash ()\ \Leftarrow \hat\alpha\ \dashv \Gamma[\hat\alpha : \star = 1]}\ 1\mathsf{I}\hat\alpha$$

$$\frac{\nu\ \mathit{chk\text{-}I} \qquad \Gamma, \alpha : \kappa \vdash \nu \Leftarrow A\ p \dashv \Delta, \alpha : \kappa, \Theta}{\Gamma \vdash \nu \Leftarrow \forall \alpha : \kappa.\ A\ p \dashv \Delta}\ \forall\mathsf{I} \qquad \frac{\Gamma, \hat\alpha : \kappa \vdash e\ s : [\hat\alpha/\alpha]A\ \gg C\ q \dashv \Delta}{\Gamma \vdash e\ s : \forall \alpha : \kappa.\ A\ p \gg C\ q \dashv \Delta}\ \forall\mathsf{Spine}$$

$$\frac{e\ \text{not a case} \qquad \Gamma \vdash P\ \mathit{true} \dashv \Theta \qquad \Theta \vdash e \Leftarrow [\Theta]A\ p \dashv \Delta}{\Gamma \vdash e \Leftarrow A \wedge P\ p \dashv \Delta}\ \wedge\mathsf{I} \qquad \frac{\nu\ \mathit{chk\text{-}I} \qquad \Gamma, \blacktriangleright_P \,/\, P \dashv \Theta \qquad \Theta \vdash \nu \Leftarrow [\Theta]A\ !\ \dashv \Delta, \blacktriangleright_P, \Delta'}{\Gamma \vdash \nu \Leftarrow P \supset A\ !\ \dashv \Delta}\ \supset\mathsf{I}$$

$$\frac{\nu\ \mathit{chk\text{-}I} \qquad \Gamma, \blacktriangleright_P \,/\, P \dashv \bot}{\Gamma \vdash \nu \Leftarrow P \supset A\ !\ \dashv \Gamma}\ \supset\mathsf{I}\bot \qquad \frac{\Gamma \vdash P\ \mathit{true} \dashv \Theta \qquad \Theta \vdash e\ s : [\Theta]A\ p \gg C\ q \dashv \Delta}{\Gamma \vdash e\ s : P \supset A\ p \gg C\ q \dashv \Delta}\ \supset\mathsf{Spine}$$

$$\frac{\Gamma, x : A\,p \vdash \nu \Leftarrow A\ p \dashv \Delta, x : A\,p, \Theta}{\Gamma \vdash \mathsf{rec}\ x.\nu \Leftarrow A\ p \dashv \Delta}\ \mathsf{Rec}$$

$$\frac{\Gamma, x : A\,p \vdash e \Leftarrow B\ p \dashv \Delta, x : A\,p, \Theta}{\Gamma \vdash \lambda x.\, e \Leftarrow A \to B\ p \dashv \Delta}\ \to\mathsf{I} \qquad \frac{\Gamma[\hat\alpha_1{:}\star, \hat\alpha_2{:}\star, \hat\alpha{:}\star = \hat\alpha_1 \to \hat\alpha_2], x : \hat\alpha_1 \vdash e \Leftarrow \hat\alpha_2 \dashv \Delta, x : \hat\alpha_1, \Delta'}{\Gamma[\hat\alpha : \star] \vdash \lambda x.\, e \Leftarrow \hat\alpha \dashv \Delta}\ \to\mathsf{I}\hat\alpha$$

$$\frac{\Gamma \vdash e \Rightarrow A\ p \dashv \Theta \qquad \Theta \vdash s : A\ p \gg C\ \lceil q \rceil \dashv \Delta}{\Gamma \vdash e\ s \Rightarrow C\ q \dashv \Delta}\ \to\mathsf{E} \qquad \frac{\Gamma \vdash s : A\ !\ \gg C \not/\!/ \dashv \Delta \qquad \mathsf{FEV}(C) = \emptyset}{\Gamma \vdash s : A\ !\ \gg C\ \lceil ! \rceil \dashv \Delta}\ \mathsf{SpineRecover} \qquad \frac{\Gamma \vdash s : A\ p \gg C\ q \dashv \Delta \qquad \big((p = \not/\!/)\ \text{ or }\ (q = !)\ \text{ or }\ (\mathsf{FEV}(C) \neq \emptyset)\big)}{\Gamma \vdash s : A\ p \gg C\ \lceil q \rceil \dashv \Delta}\ \mathsf{SpinePass}$$

$$\frac{}{\Gamma \vdash \cdot : A\ p \gg A\ p \dashv \Gamma}\ \mathsf{EmptySpine} \qquad \frac{\Gamma \vdash e \Leftarrow A\ p \dashv \Theta \qquad \Theta \vdash s : [\Theta]B\ p \gg C\ q \dashv \Delta}{\Gamma \vdash e\ s : A \to B\ p \gg C\ q \dashv \Delta}\ \to\mathsf{Spine}$$

$$\frac{\Gamma \vdash e \Leftarrow A_k\ p \dashv \Delta}{\Gamma \vdash \mathsf{inj}_k\, e \Leftarrow A_1 + A_2\ p \dashv \Delta}\ +\mathsf{I}_k \qquad \frac{\Gamma[\hat\alpha_1 : \star, \hat\alpha_2 : \star, \hat\alpha : \star = \hat\alpha_1 + \hat\alpha_2] \vdash e \Leftarrow \hat\alpha_k \dashv \Delta}{\Gamma[\hat\alpha : \star] \vdash \mathsf{inj}_k\, e \Leftarrow \hat\alpha \dashv \Delta}\ +\mathsf{I}\hat\alpha_k$$

$$\frac{\Gamma \vdash e_1 \Leftarrow A_1\ p \dashv \Theta \qquad \Theta \vdash e_2 \Leftarrow [\Theta]A_2\ p \dashv \Delta}{\Gamma \vdash \langle e_1, e_2 \rangle \Leftarrow A_1 \times A_2\ p \dashv \Delta}\ \times\mathsf{I} \qquad \frac{\Gamma[\hat\alpha_2{:}\star, \hat\alpha_1{:}\star, \hat\alpha{:}\star = \hat\alpha_1 \times \hat\alpha_2] \vdash e_1 \Leftarrow \hat\alpha_1 \dashv \Theta \qquad \Theta \vdash e_2 \Leftarrow [\Theta]\hat\alpha_2 \dashv \Delta}{\Gamma[\hat\alpha : \star] \vdash \langle e_1, e_2 \rangle \Leftarrow \hat\alpha \dashv \Delta}\ \times\mathsf{I}\hat\alpha$$

$$\frac{\Gamma \vdash t = \mathsf{zero}\ \mathit{true} \dashv \Delta}{\Gamma \vdash []\ \Leftarrow (\mathsf{Vec}\ t\ A)\ p \dashv \Delta}\ \mathsf{Nil}$$

$$\frac{\Gamma, \blacktriangleright_{\hat\alpha}, \hat\alpha : \mathbb{N} \vdash t = \mathsf{succ}(\hat\alpha)\ \mathit{true} \dashv \Gamma' \qquad \Gamma' \vdash e_1 \Leftarrow [\Gamma']A\ p \dashv \Theta \qquad \Theta \vdash e_2 \Leftarrow [\Theta](\mathsf{Vec}\ \hat\alpha\ A) \not/\!/ \dashv \Delta, \blacktriangleright_{\hat\alpha}, \Delta'}{\Gamma \vdash e_1 :: e_2 \Leftarrow (\mathsf{Vec}\ t\ A)\ p \dashv \Delta}\ \mathsf{Cons}$$

$$\frac{\Gamma[\hat\alpha_2 : \star, \hat\alpha_1 : \star, \hat\alpha : \star = \hat\alpha_1 \to \hat\alpha_2] \vdash e\ s : (\hat\alpha_1 \to \hat\alpha_2)\ \gg C\ \dashv \Delta}{\Gamma[\hat\alpha : \star] \vdash e\ s : \hat\alpha\ \gg C\ \dashv \Delta}\ \hat\alpha\mathsf{Spine} \qquad \frac{\Gamma \vdash e \Rightarrow A\ !\ \dashv \Theta \qquad \Theta \vdash \Pi :: [\Theta]A \Leftarrow [\Theta]C\ p \dashv \Delta \qquad \Delta \vdash \Pi\ \mathit{covers}\ [\Delta]A}{\Gamma \vdash \mathsf{case}(e, \Pi) \Leftarrow C\ p \dashv \Delta}\ \mathsf{Case}$$

Figure 11a. Algorithmic typing, including rules omitted from main paper

$\boxed{\Gamma \longrightarrow \Delta}$ Γ is extended by Δ

$$\frac{}{\cdot \longrightarrow \cdot}\ \longrightarrow\!\mathsf{Id} \qquad \frac{\Gamma \longrightarrow \Delta \qquad [\Delta]A = [\Delta]A'}{\Gamma, x : A\ p \longrightarrow \Delta, x : A'\ p}\ \longrightarrow\!\mathsf{Var} \qquad \frac{\Gamma \longrightarrow \Delta}{\Gamma, \alpha : \kappa \longrightarrow \Delta, \alpha : \kappa}\ \longrightarrow\!\mathsf{Uvar}$$

$$\frac{\Gamma \longrightarrow \Delta \qquad [\Delta]t = [\Delta]t'}{\Gamma, \alpha = t \longrightarrow \Delta, \alpha = t'}\ \longrightarrow\!\mathsf{Eqn} \qquad \frac{\Gamma \longrightarrow \Delta}{\Gamma, \hat{\alpha} : \kappa \longrightarrow \Delta, \hat{\alpha} : \kappa}\ \longrightarrow\!\mathsf{Unsolved} \qquad \frac{\Gamma \longrightarrow \Delta \qquad [\Delta]t = [\Delta]t'}{\Gamma, \hat{\alpha} : \kappa = t \longrightarrow \Delta, \hat{\alpha} : \kappa = t'}\ \longrightarrow\!\mathsf{Solved}$$

$$\frac{\Gamma \longrightarrow \Delta}{\Gamma, \hat{\beta} : \kappa' \longrightarrow \Delta, \hat{\beta} : \kappa' = t}\ \longrightarrow\!\mathsf{Solve} \qquad \frac{\Gamma \longrightarrow \Delta}{\Gamma \longrightarrow \Delta, \hat{\alpha} : \kappa}\ \longrightarrow\!\mathsf{Add} \qquad \frac{\Gamma \longrightarrow \Delta}{\Gamma \longrightarrow \Delta, \hat{\alpha} : \kappa = t}\ \longrightarrow\!\mathsf{AddSolved} \qquad \frac{\Gamma \longrightarrow \Delta}{\Gamma, \blacktriangleright_u \longrightarrow \Delta, \blacktriangleright_u}\ \longrightarrow\!\mathsf{Marker}$$

Figure 14. Context extension

$\boxed{\Psi \vdash \Pi :: \vec{A} \Leftarrow C\ p}$ Under context Ψ, check branches Π with patterns of type $\vec{A}$ and bodies of type C

$$\frac{}{\Psi \vdash \cdot :: \vec{A} \Leftarrow C\ p}\ \mathsf{DeclMatchEmpty} \qquad \frac{\Psi \vdash \pi :: \vec{A} \Leftarrow C\ p \qquad \Psi \vdash \Pi :: \vec{A} \Leftarrow C\ p}{\Psi \vdash (\pi \mid \Pi) :: \vec{A} \Leftarrow C\ p}\ \mathsf{DeclMatchSeq}$$

$$\frac{\Psi \vdash e \Leftarrow C\ p}{\Psi \vdash (\cdot \Rightarrow e) :: \cdot \Leftarrow C\ p}\ \mathsf{DeclMatchBase} \qquad \frac{\Psi \vdash \vec{\rho} \Rightarrow e :: \vec{A} \Leftarrow C\ p}{\Psi \vdash (), \vec{\rho} \Rightarrow e :: 1, \vec{A} \Leftarrow C\ p}\ \mathsf{DeclMatchUnit}$$

$$\frac{\Psi, \alpha : \kappa \vdash \vec{\rho} \Rightarrow e :: A, \vec{A} \Leftarrow C\ p}{\Psi \vdash (\vec{\rho} \Rightarrow e) :: (\exists \alpha : \kappa.\ A), \vec{A} \Leftarrow C\ p}\ \mathsf{DeclMatch}\exists \qquad \frac{\Psi \vdash \rho_1, \rho_2, \vec{\rho} \Rightarrow e :: A_1, A_2, \vec{A} \Leftarrow C\ p}{\Psi \vdash \langle \rho_1, \rho_2 \rangle, \vec{\rho} \Rightarrow e :: (A_1 \times A_2), \vec{A} \Leftarrow C\ p}\ \mathsf{DeclMatch}\times$$

$$\frac{\Psi \vdash \rho, \vec{\rho} \Rightarrow e :: A_k, \vec{A} \Leftarrow C\ p}{\Psi \vdash \mathsf{inj}_k\, \rho, \vec{\rho} \Rightarrow e :: A_1 + A_2, \vec{A} \Leftarrow C\ p}\ \mathsf{DeclMatch}+_k \qquad \frac{\Psi \,/\, P \vdash \vec{\rho} \Rightarrow e :: A, \vec{A} \Leftarrow C\ p}{\Psi \vdash \vec{\rho} \Rightarrow e :: (A \wedge P), \vec{A} \Leftarrow C\ p}\ \mathsf{DeclMatch}\wedge$$

$$\frac{\Psi \,/\, (t = \mathsf{zero}) \vdash \vec{\rho} \Rightarrow e :: \vec{A} \Leftarrow C\ p}{\Psi \vdash [], \vec{\rho} \Rightarrow e :: (\mathsf{Vec}\ t\ A), \vec{A} \Leftarrow C\ p}\ \mathsf{DeclMatchNil}$$

$$\frac{\Psi, \alpha : \mathbb{N} \,/\, (t = \mathsf{succ}(\alpha)) \vdash \rho_1, \rho_2, \vec{\rho} \Rightarrow e :: A, (\mathsf{Vec}\ \alpha\ A), \vec{A} \Leftarrow C\ p}{\Psi \vdash (\rho_1 :: \rho_2), \vec{\rho} \Rightarrow e :: (\mathsf{Vec}\ t\ A), \vec{A} \Leftarrow C\ p}\ \mathsf{DeclMatchCons}$$

$$\frac{A \text{ not headed by } \wedge \text{ or } \exists \qquad \Psi, x : A\,! \vdash \vec{\rho} \Rightarrow e :: \vec{A} \Leftarrow C\ p}{\Psi \vdash x, \vec{\rho} \Rightarrow e :: A, \vec{A} \Leftarrow C\ p}\ \mathsf{DeclMatchNeg} \qquad \frac{A \text{ not headed by } \wedge \text{ or } \exists \qquad \Psi \vdash \vec{\rho} \Rightarrow e :: \vec{A} \Leftarrow C\ p}{\Psi \vdash \_, \vec{\rho} \Rightarrow e :: A, \vec{A} \Leftarrow C\ p}\ \mathsf{DeclMatchWild}$$

$\boxed{\Psi \,/\, P \vdash \Pi :: \vec{A} \Leftarrow C\ p}$ Under context Ψ, incorporate proposition P while checking branches Π with patterns of type $\vec{A}$ and bodies of type C

$$\frac{\mathsf{mgu}(\sigma, \tau) = \bot}{\Psi \,/\, \sigma = \tau \vdash \vec{\rho} \Rightarrow e :: \vec{A} \Leftarrow C\ p}\ \mathsf{DeclMatch}\bot \qquad \frac{\mathsf{mgu}(\sigma, \tau) = \theta \qquad \theta(\Psi) \vdash \theta(\vec{\rho} \Rightarrow e) :: \theta(\vec{A}) \Leftarrow \theta(C)\ p}{\Psi \,/\, \sigma = \tau \vdash \vec{\rho} \Rightarrow e :: \vec{A} \Leftarrow C\ p}\ \mathsf{DeclMatchUnify}$$

Figure 15. Declarative pattern matching

$\boxed{\Psi \vdash \Pi \textit{ covers } \vec{A}}$ Patterns Π cover the types $\vec{A}$ in context Ψ

$\boxed{\Psi \;/\; P \vdash \Pi \textit{ covers } \vec{A}}$ Patterns Π cover the types $\vec{A}$ in context Ψ, assuming P

$$\frac{}{\Psi \vdash (\cdot \Rightarrow e_1) \mid \Pi' \textit{ covers } \cdot}\ \textsf{DeclCoversEmpty} \qquad \frac{\Pi \overset{\text{var}}{\rightsquigarrow} \Pi' \qquad \Psi \vdash \Pi' \textit{ covers } \vec{A}}{\Psi \vdash \Pi \textit{ covers } A, \vec{A}}\ \textsf{DeclCoversVar}$$

$$\frac{\Pi \overset{1}{\rightsquigarrow} \Pi' \qquad \Psi \vdash \Pi' \textit{ covers } \vec{A}}{\Psi \vdash \Pi \textit{ covers } 1, \vec{A}}\ \textsf{DeclCovers1} \qquad \frac{\Pi \overset{\times}{\rightsquigarrow} \Pi' \qquad \Psi \vdash \Pi' \textit{ covers } A_1, A_2, \vec{A}}{\Psi \vdash \Pi \textit{ covers } (A_1 \times A_2), \vec{A}}\ \textsf{DeclCovers}\times$$

$$\frac{\Pi \overset{+}{\rightsquigarrow} \Pi_L \parallel \Pi_R \qquad \Psi \vdash \Pi_L \textit{ covers } A_1, \vec{A} \qquad \Psi \vdash \Pi_R \textit{ covers } A_2, \vec{A}}{\Psi \vdash \Pi \textit{ covers } (A_1 + A_2), \vec{A}}\ \textsf{DeclCovers}+ \qquad \frac{\Psi, \alpha : \kappa \vdash \Pi \textit{ covers } A, \vec{A}}{\Psi \vdash \Pi \textit{ covers } (\exists \alpha : \kappa.\, A), \vec{A}}\ \textsf{DeclCovers}\exists$$

$$\frac{\Psi \;/\; t_1 = t_2 \vdash \Pi \textit{ covers } A_0, \vec{A}}{\Psi \vdash \Pi \textit{ covers } \big(A_0 \wedge (t_1 = t_2)\big), \vec{A}}\ \textsf{DeclCovers}\wedge$$

$$\frac{\Pi \overset{\text{Vec}}{\rightsquigarrow} \Pi_{[]} \parallel \Pi_{::} \qquad \Psi \;/\; t = \mathsf{zero} \vdash \Pi_{[]} \textit{ covers } \vec{A} \qquad \Psi, n : \mathbb{N} \;/\; t = \mathsf{succ}(n) \vdash \Pi_{::} \textit{ covers } (A, \mathsf{Vec}\ n\ A, \vec{A})}{\Psi \vdash \Pi \textit{ covers } \vec{t}A, \vec{A}}\ \textsf{DeclCoversVec}$$

$$\frac{\theta = \mathsf{mgu}(t_1, t_2) \qquad \theta(\Psi) \vdash \theta(\Pi) \textit{ covers } \theta(\vec{A})}{\Psi \;/\; t_1 = t_2 \vdash \Pi \textit{ covers } \vec{A}}\ \textsf{DeclCoversEq} \qquad \frac{\mathsf{mgu}(t_1, t_2) = \bot}{\Psi \;/\; t_1 = t_2 \vdash \Pi \textit{ covers } \vec{A}}\ \textsf{DeclCoversEqBot}$$

$\boxed{\Pi \overset{\text{Vec}}{\rightsquigarrow} \Pi_{[]} \parallel \Pi_{::}}$ Expand vector patterns in Π

$$\frac{}{\cdot \overset{\text{Vec}}{\rightsquigarrow} \cdot \parallel \cdot} \qquad \frac{\rho \in \{x, \_\} \qquad \Pi \overset{\text{Vec}}{\rightsquigarrow} \Pi_{[]} \parallel \Pi_{::}}{(\rho, \vec{\rho} \Rightarrow e) \mid \Pi \overset{\text{Vec}}{\rightsquigarrow} (\_, \vec{\rho} \Rightarrow e) \mid \Pi_{[]} \parallel (\_, \vec{\rho} \Rightarrow e) \mid \Pi_{::}} \qquad \frac{\Pi \overset{\text{Vec}}{\rightsquigarrow} \Pi_{[]} \parallel \Pi_{::}}{([], \vec{\rho} \Rightarrow e) \mid \Pi \overset{\text{Vec}}{\rightsquigarrow} (\vec{\rho} \Rightarrow e) \mid \Pi_{[]} \parallel \Pi_{::}}$$

$$\frac{\Pi \overset{\text{Vec}}{\rightsquigarrow} \Pi_{[]} \parallel \Pi_{::}}{((\rho :: \rho', \vec{\rho} \Rightarrow e) \mid \Pi \overset{\text{Vec}}{\rightsquigarrow} \Pi_{[]} \parallel (\rho, \rho', \vec{\rho} \Rightarrow e) \mid \Pi_{::}}$$

$\boxed{\Pi \overset{\times}{\rightsquigarrow} \Pi'}$ Expand head pair patterns in Π

$$\frac{}{\cdot \overset{\times}{\rightsquigarrow} \cdot} \qquad \frac{\Pi \overset{\times}{\rightsquigarrow} \Pi'}{(\langle \rho_1, \rho_2 \rangle, \vec{\rho} \Rightarrow e) \mid \Pi \overset{\times}{\rightsquigarrow} (\rho_1, \rho_2, \vec{\rho} \Rightarrow e) \mid \Pi'} \qquad \frac{\rho \in \{z, \_\} \qquad \Pi \overset{\times}{\rightsquigarrow} \Pi'}{(\rho, \vec{\rho} \Rightarrow e) \mid \Pi \overset{\times}{\rightsquigarrow} (\_, \_, \vec{\rho} \Rightarrow e) \mid \Pi'}$$

$\boxed{\Pi \overset{+}{\rightsquigarrow} \Pi_L \parallel \Pi_R}$ Expand head sum patterns in Π into left Π_L and right Π_R sets

$$\frac{}{\cdot \overset{+}{\rightsquigarrow} \cdot \parallel \cdot} \qquad \frac{\rho \in \{x, \_\} \qquad \Pi \overset{+}{\rightsquigarrow} \Pi_L \parallel \Pi_R}{(\rho, \vec{\rho} \Rightarrow e) \mid \Pi \overset{+}{\rightsquigarrow} (\_, \vec{\rho} \Rightarrow e) \mid \Pi_L \parallel (\_, \vec{\rho} \Rightarrow e) \mid \Pi_R} \qquad \frac{\Pi \overset{+}{\rightsquigarrow} \Pi_L \parallel \Pi_R}{(\mathsf{inj}_1\ \rho, \vec{\rho} \Rightarrow e) \mid \Pi \overset{+}{\rightsquigarrow} (\rho, \vec{\rho} \Rightarrow e) \mid \Pi_L \parallel \Pi_R}$$

$$\frac{\Pi \overset{+}{\rightsquigarrow} \Pi_L \parallel \Pi_R}{(\mathsf{inj}_2\ \rho, \vec{\rho} \Rightarrow e) \mid \Pi \overset{+}{\rightsquigarrow} \Pi_L \parallel (\rho, \vec{\rho} \Rightarrow e) \mid \Pi_R}$$

$\boxed{\Pi \overset{\text{var}}{\rightsquigarrow} \Pi'}$ Remove head variable and wildcard patterns from Π

$$\frac{}{\cdot \overset{\text{var}}{\rightsquigarrow} \cdot} \qquad \frac{\rho \in \{x, \_\} \qquad \Pi \overset{\text{var}}{\rightsquigarrow} \Pi'}{(\rho, \vec{\rho} \Rightarrow e) \mid \Pi \overset{\text{var}}{\rightsquigarrow} (\vec{\rho} \Rightarrow e) \mid \Pi'}$$

$\boxed{\Pi \overset{1}{\rightsquigarrow} \Pi'}$ Remove head variable, wildcard, and unit patterns from Π

$$\frac{}{\cdot \overset{1}{\rightsquigarrow} \cdot} \qquad \frac{\rho \in \{x, \_, ()\} \qquad \Pi \overset{1}{\rightsquigarrow} \Pi'}{(\rho, \vec{\rho} \Rightarrow e) \mid \Pi \overset{1}{\rightsquigarrow} (\vec{\rho} \Rightarrow e) \mid \Pi'}$$

Figure 16. Match coverage

$\boxed{\Psi \vdash t : \kappa}$ Under context Ψ, term t has sort κ

$$\frac{(\alpha : \kappa) \in \Psi}{\Psi \vdash \alpha : \kappa}\ \mathsf{UvarSort} \qquad \frac{}{\Psi \vdash 1 : \star}\ \mathsf{UnitSort} \qquad \frac{\Psi \vdash t_1 : \star \qquad \Psi \vdash t_2 : \star}{\Psi \vdash t_1 \oplus t_2 : \star}\ \mathsf{BinSort} \qquad \frac{}{\Psi \vdash \mathsf{zero} : \mathbb{N}}\ \mathsf{ZeroSort} \qquad \frac{\Psi \vdash t : \mathbb{N}}{\Psi \vdash \mathsf{succ}(t) : \mathbb{N}}\ \mathsf{SuccSort}$$

$\boxed{\Psi \vdash P\ \mathit{prop}}$ Under context Ψ, proposition P is well-formed

$$\frac{\Psi \vdash t : \mathbb{N} \qquad \Psi \vdash t' : \mathbb{N}}{\Psi \vdash t = t'\ \mathit{prop}}\ \mathsf{EqDeclProp}$$

$\boxed{\Psi \vdash A\ \mathit{type}}$ Under context Ψ, type A is well-formed

$$\frac{(\alpha : \star) \in \Psi}{\Psi \vdash \alpha\ \mathit{type}}\ \mathsf{DeclUvarWF} \qquad \frac{}{\Psi \vdash 1\ \mathit{type}}\ \mathsf{DeclUnitWF} \qquad \frac{\Psi \vdash A\ \mathit{type} \qquad \Psi \vdash B\ \mathit{type} \qquad \oplus \in \{\rightarrow, \times, +\}}{\Psi \vdash A \oplus B\ \mathit{type}}\ \mathsf{DeclBinWF}$$

$$\frac{\Gamma \vdash t : \mathbb{N} \qquad \Gamma \vdash A\ \mathit{type}}{\Gamma \vdash \mathsf{Vec}\ t\ A\ \mathit{type}}\ \mathsf{DeclVecWF} \qquad \frac{\Psi, \alpha : \kappa \vdash A\ \mathit{type}}{\Psi \vdash (\forall \alpha : \kappa.\ A)\ \mathit{type}}\ \mathsf{DeclAllWF} \qquad \frac{\Psi, \alpha : \kappa \vdash A\ \mathit{type}}{\Psi \vdash (\exists \alpha : \kappa.\ A)\ \mathit{type}}\ \mathsf{DeclExistsWF}$$

$$\frac{\Psi \vdash P\ \mathit{prop} \qquad \Psi \vdash A\ \mathit{type}}{\Psi \vdash P \supset A\ \mathit{type}}\ \mathsf{DeclImpliesWF} \qquad \frac{\Psi \vdash P\ \mathit{prop} \qquad \Psi \vdash A\ \mathit{type}}{\Psi \vdash A \wedge P\ \mathit{type}}\ \mathsf{DeclWithWF}$$

$\boxed{\Psi \vdash \vec{A}\ \mathit{types}}$ Under context Ψ, types in $\vec{A}$ are well-formed

$$\frac{\text{for all } A \in \vec{A}.\ \ \Psi \vdash A\ \mathit{type}}{\Psi \vdash \vec{A}\ \mathit{types}}\ \mathsf{DeclTypevecWF}$$

$\boxed{\Psi\ \mathit{ctx}}$ Declarative context Ψ is well-formed

$$\frac{}{\cdot\ \mathit{ctx}}\ \mathsf{EmptyDeclCtx} \qquad \frac{\Psi\ \mathit{ctx} \qquad x \notin \mathsf{dom}(\Psi) \qquad \Psi \vdash A\ \mathit{type}}{\Psi, x : A\ \mathit{ctx}}\ \mathsf{HypDeclCtx} \qquad \frac{\Psi\ \mathit{ctx} \qquad \alpha \notin \mathsf{dom}(\Psi)}{\Psi, \alpha : \kappa\ \mathit{ctx}}\ \mathsf{VarDeclCtx}$$

Figure 17. Sorting; well-formedness of propositions, types, and contexts in the declarative system

$\boxed{\Gamma \vdash \tau : \kappa}$ Under context Γ, term τ has sort κ

$$\frac{(u : \kappa) \in \Gamma}{\Gamma \vdash u : \kappa}\ \mathsf{VarSort} \qquad \frac{(\hat{\alpha} : \kappa = \tau) \in \Gamma}{\Gamma \vdash \hat{\alpha} : \kappa}\ \mathsf{SolvedVarSort} \qquad \frac{}{\Gamma \vdash 1 : \star}\ \mathsf{UnitSort} \qquad \frac{\Gamma \vdash \tau_1 : \star \qquad \Gamma \vdash \tau_2 : \star}{\Gamma \vdash \tau_1 \oplus \tau_2 : \star}\ \mathsf{BinSort}$$

$$\frac{}{\Gamma \vdash \mathsf{zero} : \mathbb{N}}\ \mathsf{ZeroSort} \qquad \frac{\Gamma \vdash t : \mathbb{N}}{\Gamma \vdash \mathsf{succ}(t) : \mathbb{N}}\ \mathsf{SuccSort}$$

$\boxed{\Gamma \vdash P\ \mathit{prop}}$ Under context Γ, proposition P is well-formed

$$\frac{\Gamma \vdash t : \mathbb{N} \qquad \Gamma \vdash t' : \mathbb{N}}{\Gamma \vdash t = t'\ \mathit{prop}}\ \mathsf{EqProp}$$

$\boxed{\Gamma \vdash A\ \mathit{type}}$ Under context Γ, type A is well-formed

$$\frac{(u : \star) \in \Gamma}{\Gamma \vdash u\ \mathit{type}}\ \mathsf{VarWF} \qquad \frac{(\hat{\alpha} : \star = \tau) \in \Gamma}{\Gamma \vdash \hat{\alpha}\ \mathit{type}}\ \mathsf{SolvedVarWF} \qquad \frac{}{\Gamma \vdash 1\ \mathit{type}}\ \mathsf{UnitWF}$$

$$\frac{\Gamma \vdash A\ \mathit{type} \qquad \Gamma \vdash B\ \mathit{type} \qquad \oplus \in \{\rightarrow, \times, +\}}{\Gamma \vdash A \oplus B\ \mathit{type}}\ \mathsf{BinWF} \qquad \frac{\Gamma \vdash t : \mathbb{N} \qquad \Gamma \vdash A\ \mathit{type}}{\Gamma \vdash \mathsf{Vec}\ t\ A\ \mathit{type}}\ \mathsf{VecWF} \qquad \frac{\Gamma, \alpha : \kappa \vdash A\ \mathit{type}}{\Gamma \vdash \forall \alpha : \kappa.\ A\ \mathit{type}}\ \mathsf{ForallWF}$$

$$\frac{\Gamma, \alpha : \kappa \vdash A\ \mathit{type}}{\Gamma \vdash \exists \alpha : \kappa.\ A\ \mathit{type}}\ \mathsf{ExistsWF} \qquad \frac{\Gamma \vdash P\ \mathit{prop} \qquad \Gamma \vdash A\ \mathit{type}}{\Gamma \vdash P \supset A\ \mathit{type}}\ \mathsf{ImpliesWF} \qquad \frac{\Gamma \vdash P\ \mathit{prop} \qquad \Gamma \vdash A\ \mathit{type}}{\Gamma \vdash A \wedge P\ \mathit{type}}\ \mathsf{WithWF}$$

$\boxed{\Gamma \vdash A\ p\ \mathit{type}}$ Under context Γ, type A is well-formed and respects principality p

$$\frac{\Gamma \vdash A\ \mathit{type} \qquad \mathsf{FEV}([\Gamma]A) = \emptyset}{\Gamma \vdash A\ !\ \mathit{type}}\ \mathsf{PrincipalWF} \qquad \frac{\Gamma \vdash A\ \mathit{type}}{\Gamma \vdash A \not!\ \mathit{type}}\ \mathsf{NonPrincipalWF}$$

$\boxed{\Gamma \vdash \vec{A}\ [p]\ \mathit{types}}$ Under context Γ, types in $\vec{A}$ are well-formed [with principality p]

$$\frac{\text{for all } A \in \vec{A}.\ \ \Gamma \vdash A\ \mathit{type}}{\Gamma \vdash \vec{A}\ \mathit{types}}\ \mathsf{TypevecWF} \qquad \frac{\text{for all } A \in \vec{A}.\ \ \Gamma \vdash A\ p\ \mathit{type}}{\Gamma \vdash \vec{A}\ p\ \mathit{types}}\ \mathsf{PrincipalTypevecWF}$$

$\boxed{\Gamma\ \mathit{ctx}}$ Algorithmic context Γ is well-formed

$$\frac{}{\cdot\ \mathit{ctx}}\ \mathsf{EmptyCtx} \qquad \frac{\Gamma\ \mathit{ctx} \qquad x \notin \mathsf{dom}(\Gamma) \qquad \Gamma \vdash A\ \mathit{type}}{\Gamma, x : A \not!\ \mathit{ctx}}\ \mathsf{HypCtx} \qquad \frac{\Gamma\ \mathit{ctx} \qquad x \notin \mathsf{dom}(\Gamma) \qquad \Gamma \vdash A\ \mathit{type} \qquad \mathsf{FEV}([\Gamma]A) = \emptyset}{\Gamma, x : A\ !\ \mathit{ctx}}\ \mathsf{Hyp!Ctx}$$

$$\frac{\Gamma\ \mathit{ctx} \qquad u \notin \mathsf{dom}(\Gamma)}{\Gamma, u : \kappa\ \mathit{ctx}}\ \mathsf{VarCtx} \qquad \frac{\Gamma\ \mathit{ctx} \qquad \hat{\alpha} \notin \mathsf{dom}(\Gamma) \qquad \Gamma \vdash t : \kappa}{\Gamma, \hat{\alpha} : \kappa = t\ \mathit{ctx}}\ \mathsf{SolvedCtx}$$

$$\frac{\Gamma\ \mathit{ctx} \qquad \alpha : \kappa \in \Gamma \qquad (\alpha = -) \notin \Gamma \qquad \Gamma \vdash \tau : \kappa}{\Gamma, \alpha = \tau\ \mathit{ctx}}\ \mathsf{EqnVarCtx} \qquad \frac{\Gamma\ \mathit{ctx} \qquad \blacktriangleright_u \notin \Gamma}{\Gamma, \blacktriangleright_u\ \mathit{ctx}}\ \mathsf{MarkerCtx}$$

Figure 18. Well-formedness of types and contexts in the algorithmic system

$\boxed{\Gamma \vdash\ \mathsf{P}\ \mathit{true}\ \dashv \Delta}$ Under context Γ, check P, with output context Δ

$$\frac{\Gamma \vdash\ \mathsf{t}_1 \doteq \mathsf{t}_2 : \mathbb{N} \dashv \Delta}{\Gamma \vdash\ \mathsf{t}_1 = \mathsf{t}_2\ \mathit{true} \dashv \Delta}\ \mathsf{CheckpropEq}$$

$\boxed{\Gamma\ /\ \mathsf{P} \dashv \Delta^{\perp}}$ Incorporate hypothesis P into Γ, producing Δ or inconsistency $\perp$

$$\frac{\Gamma\ /\ \mathsf{t}_1 \doteq \mathsf{t}_2 : \mathbb{N} \dashv \Delta^{\perp}}{\Gamma\ /\ \mathsf{t}_1 = \mathsf{t}_2 \dashv \Delta^{\perp}}\ \mathsf{ElimpropEq}$$

Figure 19. Checking and assuming propositions

$\boxed{\Gamma \vdash\ \mathsf{t}_1 \doteq \mathsf{t}_2 : \kappa \dashv \Delta}$ Check that t_1 equals t_2, taking Γ to Δ

$$\frac{}{\Gamma \vdash\ \mathsf{u} \doteq \mathsf{u} : \kappa \dashv \Gamma}\ \mathsf{CheckeqVar} \qquad \frac{}{\Gamma \vdash\ 1 \doteq 1 : \star \dashv \Gamma}\ \mathsf{CheckeqUnit}$$

$$\frac{\Gamma \vdash\ \tau_1 \doteq \tau_1' : \star \dashv \Theta \qquad \Theta \vdash\ [\Theta]\tau_2 \doteq [\Theta]\tau_2' : \star \dashv \Delta}{\Gamma \vdash\ (\tau_1 \oplus \tau_2) \doteq (\tau_1' \oplus \tau_2') : \star \dashv \Delta}\ \mathsf{CheckeqBin}$$

$$\frac{}{\Gamma \vdash\ \mathsf{zero} \doteq \mathsf{zero} : \mathbb{N} \dashv \Gamma}\ \mathsf{CheckeqZero} \qquad \frac{\Gamma \vdash\ \mathsf{t}_1 \doteq \mathsf{t}_2 : \mathbb{N} \dashv \Delta}{\Gamma \vdash\ \mathsf{succ}(\mathsf{t}_1) \doteq \mathsf{succ}(\mathsf{t}_2) : \mathbb{N} \dashv \Delta}\ \mathsf{CheckeqSucc}$$

$$\frac{\Gamma[\hat{\alpha} : \kappa] \vdash\ \hat{\alpha} \coloneqq \mathsf{t} : \kappa \dashv \Delta \qquad \hat{\alpha} \notin \mathrm{FV}(\mathsf{t})}{\Gamma[\hat{\alpha} : \kappa] \vdash\ \hat{\alpha} \doteq \mathsf{t} : \kappa \dashv \Delta}\ \mathsf{CheckeqInstL} \qquad \frac{\Gamma[\hat{\alpha} : \kappa] \vdash\ \hat{\alpha} \coloneqq \mathsf{t} : \kappa \dashv \Delta \qquad \hat{\alpha} \notin \mathrm{FV}(\mathsf{t})}{\Gamma[\hat{\alpha} : \kappa] \vdash\ \mathsf{t} \doteq \hat{\alpha} : \kappa \dashv \Delta}\ \mathsf{CheckeqInstR}$$

Figure 20. Checking equations

$\boxed{\mathsf{t}_1\ \#\ \mathsf{t}_2}$ t_1 and t_2 have incompatible head constructors

$$\frac{}{\mathsf{zero}\ \#\ \mathsf{succ}(\mathsf{t})} \qquad \frac{}{\mathsf{succ}(\mathsf{t})\ \#\ \mathsf{zero}} \qquad \frac{}{1\ \#\ (\tau_1 \oplus \tau_2)} \qquad \frac{}{(\tau_1 \oplus \tau_2)\ \#\ 1} \qquad \frac{\oplus_1 \neq \oplus_2}{(\sigma_1 \oplus_1 \tau_1)\ \#\ (\sigma_2 \oplus_2 \tau_2)}$$

Figure 21. Head constructor clash

$\boxed{\Gamma\ /\ \sigma \doteq \tau : \kappa \dashv \Delta^{\perp}}$ Unify σ and τ, taking Γ to Δ, or to inconsistency $\perp$

$$\frac{}{\Gamma\ /\ \alpha \doteq \alpha : \kappa \dashv \Gamma}\ \mathsf{ElimeqUvarRefl}$$

$$\frac{}{\Gamma\ /\ \mathsf{zero} \doteq \mathsf{zero} : \mathbb{N} \dashv \Gamma}\ \mathsf{ElimeqZero} \qquad \frac{\Gamma\ /\ \sigma \doteq \tau : \mathbb{N} \dashv \Delta^{\perp}}{\Gamma\ /\ \mathsf{succ}(\sigma) \doteq \mathsf{succ}(\tau) : \mathbb{N} \dashv \Delta^{\perp}}\ \mathsf{ElimeqSucc}$$

$$\frac{\alpha \notin \mathrm{FV}(\tau) \qquad (\alpha = -) \notin \Gamma}{\Gamma\ /\ \alpha \doteq \tau : \kappa \dashv \Gamma, \alpha = \tau}\ \mathsf{ElimeqUvarL} \qquad \frac{\alpha \notin \mathrm{FV}(\tau) \qquad (\alpha = -) \notin \Gamma}{\Gamma\ /\ \tau \doteq \alpha : \kappa \dashv \Gamma, \alpha = \tau}\ \mathsf{ElimeqUvarR} \qquad \frac{\mathsf{t} \neq \alpha \qquad \alpha \in \mathrm{FV}(\tau)}{\Gamma\ /\ \alpha \doteq \tau : \kappa \dashv \bot}\ \mathsf{ElimeqUvarL}\bot$$

$$\frac{\mathsf{t} \neq \alpha \qquad \alpha \in \mathrm{FV}(\tau)}{\Gamma\ /\ \tau \doteq \alpha : \kappa \dashv \bot}\ \mathsf{ElimeqUvarR}\bot$$

$$\frac{}{\Gamma\ /\ 1 \doteq 1 : \star \dashv \Gamma}\ \mathsf{ElimeqUnit} \qquad \frac{\Gamma\ /\ \tau_1 \doteq \tau_1' : \star \dashv \Theta \qquad \Theta\ /\ [\Theta]\tau_2 \doteq [\Theta]\tau_2' : \star \dashv \Delta^{\perp}}{\Gamma\ /\ (\tau_1 \oplus \tau_2) \doteq (\tau_1' \oplus \tau_2') : \star \dashv \Delta^{\perp}}\ \mathsf{ElimeqBin}$$

$$\frac{\Gamma\ /\ \tau_1 \doteq \tau_1' : \star \dashv \bot}{\Gamma\ /\ (\tau_1 \oplus \tau_2) \doteq (\tau_1' \oplus \tau_2') : \star \dashv \bot}\ \mathsf{ElimeqBinBot}$$

$$\frac{\sigma\ \#\ \tau}{\Gamma\ /\ \sigma \doteq \tau : \kappa \dashv \bot}\ \mathsf{ElimeqClash}$$

Figure 22. Eliminating equations

$\boxed{\Gamma \vdash A <:^{\pm} B \dashv \Delta}$ Under input context Γ, type A is a subtype of B, with output context Δ

$$\frac{A \text{ not headed by } \forall/\exists \quad B \text{ not headed by } \forall/\exists \qquad \Gamma \vdash A \equiv B \dashv \Delta}{\Gamma \vdash A <:^{\pm} B \dashv \Delta} <:\mathsf{Equiv}$$

B not headed by $\forall$

$$\frac{\Gamma, \blacktriangleright_{\hat{\alpha}}, \hat{\alpha} : \kappa \vdash [\hat{\alpha}/\alpha]A <:^{-} B \dashv \Delta, \blacktriangleright_{\hat{\alpha}}, \Theta}{\Gamma \vdash \forall \alpha : \kappa.\, A <:^{-} B \dashv \Delta} <:\forall L \qquad \frac{\Gamma, \beta : \kappa \vdash A <:^{-} B \dashv \Delta, \beta : \kappa, \Theta}{\Gamma \vdash A <:^{-} \forall \beta : \kappa.\, B \dashv \Delta} <:\forall R$$

A not headed by $\exists$

$$\frac{\Gamma, \alpha : \kappa \vdash A <:^{+} B \dashv \Delta, \alpha : \kappa, \Theta}{\Gamma \vdash \exists \alpha : \kappa.\, A <:^{+} B \dashv \Delta} <:\exists L \qquad \frac{\Gamma, \blacktriangleright_{\hat{\beta}}, \hat{\beta} : \kappa \vdash A <:^{+} [\hat{\beta}/\beta]B \dashv \Delta, \blacktriangleright_{\hat{\beta}}, \Theta}{\Gamma \vdash A <:^{+} \exists \beta : \kappa.\, B \dashv \Delta} <:\exists R$$

$$\frac{\Gamma \vdash A <:^{-} B \dashv \Delta \qquad \mathit{neg}(A) \quad \mathit{nonpos}(B)}{\Gamma \vdash A <:^{+} B \dashv \Delta} <:^{\mp}_{+} L \qquad \frac{\Gamma \vdash A <:^{-} B \dashv \Delta \qquad \mathit{nonpos}(A) \quad \mathit{neg}(B)}{\Gamma \vdash A <:^{+} B \dashv \Delta} <:^{\mp}_{+} R$$

$$\frac{\Gamma \vdash A <:^{+} B \dashv \Delta \qquad \mathit{pos}(A) \quad \mathit{nonneg}(B)}{\Gamma \vdash A <:^{-} B \dashv \Delta} <:^{\pm}_{-} L \qquad \frac{\Gamma \vdash A <:^{+} B \dashv \Delta \qquad \mathit{nonneg}(A) \quad \mathit{pos}(B)}{\Gamma \vdash A <:^{-} B \dashv \Delta} <:^{\pm}_{-} R$$

$\boxed{\Gamma \vdash P \equiv Q \dashv \Delta}$ Under input context Γ, check that P is equivalent to Q with output context Δ

$$\frac{\Gamma \vdash t_1 \doteq t_2 : \mathbb{N} \dashv \Theta \qquad \Theta \vdash [\Theta]t_1' \doteq [\Theta]t_2' : \mathbb{N} \dashv \Delta}{\Gamma \vdash (t_1 = t_1') \equiv (t_2 = t_2') \dashv \Delta} \equiv\mathsf{PropEq}$$

$\boxed{\Gamma \vdash A \equiv B \dashv \Delta}$ Under input context Γ, check that A is equivalent to B with output context Δ

$$\frac{}{\Gamma \vdash \alpha \equiv \alpha \dashv \Gamma} \equiv\mathsf{Var} \qquad \frac{}{\Gamma \vdash \hat{\alpha} \equiv \hat{\alpha} \dashv \Gamma} \equiv\mathsf{Exvar} \qquad \frac{}{\Gamma \vdash 1 \equiv 1 \dashv \Gamma} \equiv\mathsf{Unit}$$

$$\frac{\Gamma \vdash A_1 \equiv B_1 \dashv \Theta \qquad \Theta \vdash [\Theta]A_2 \equiv [\Theta]B_2 \dashv \Delta}{\Gamma \vdash (A_1 \oplus A_2) \equiv (B_1 \oplus B_2) \dashv \Delta} \equiv\oplus \qquad \frac{\Gamma \vdash t_1 \equiv t_2 \dashv \Theta \qquad \Theta \vdash [\Theta]A_1 \equiv [\Theta]A_2 \dashv \Delta}{\Gamma \vdash (\mathsf{Vec}\ t_1\ A_1) \equiv (\mathsf{Vec}\ t_2\ A_2) \dashv \Delta} \equiv\mathsf{Vec}$$

$$\frac{\Gamma, \alpha : \kappa \vdash A \equiv B \dashv \Delta, \alpha : \kappa, \Delta'}{\Gamma \vdash (\forall \alpha : \kappa.\, A) \equiv (\forall \alpha : \kappa.\, B) \dashv \Delta} \equiv\forall \qquad \frac{\Gamma, \alpha : \kappa \vdash A \equiv B \dashv \Delta, \alpha : \kappa, \Delta'}{\Gamma \vdash (\exists \alpha : \kappa.\, A) \equiv (\exists \alpha : \kappa.\, B) \dashv \Delta} \equiv\exists$$

$$\frac{\Gamma \vdash P \equiv Q \dashv \Theta \qquad \Theta \vdash [\Theta]A \equiv [\Theta]B \dashv \Delta}{\Gamma \vdash (P \supset A) \equiv (Q \supset B) \dashv \Delta} \equiv\supset \qquad \frac{\Gamma \vdash P \equiv Q \dashv \Theta \qquad \Theta \vdash [\Theta]A \equiv [\Theta]B \dashv \Delta}{\Gamma \vdash (A \wedge P) \equiv (B \wedge Q) \dashv \Delta} \equiv\wedge$$

$$\frac{\hat{\alpha} \notin \mathrm{FV}(\tau) \qquad \Gamma[\hat{\alpha}] \vdash \hat{\alpha} \coloneqq \tau : \star \dashv \Delta}{\Gamma[\hat{\alpha}] \vdash \hat{\alpha} \equiv \tau \dashv \Delta} \equiv\mathsf{InstantiateL} \qquad \frac{\hat{\alpha} \notin \mathrm{FV}(\tau) \qquad \Gamma[\hat{\alpha}] \vdash \hat{\alpha} \coloneqq \tau : \star \dashv \Delta}{\Gamma[\hat{\alpha}] \vdash \tau \equiv \hat{\alpha} \dashv \Delta} \equiv\mathsf{InstantiateR}$$

Figure 23. Algorithmic subtyping and equivalence

$\boxed{\Gamma \vdash \hat{\alpha} \coloneqq t : \kappa \dashv \Delta}$ Under input context Γ, instantiate $\hat{\alpha}$ such that $\hat{\alpha} = t$ with output context Δ

$$\frac{\Gamma_0 \vdash \tau : \kappa}{\Gamma_0, \hat{\alpha} : \kappa, \Gamma_1 \vdash \hat{\alpha} \coloneqq \tau : \kappa \dashv \Gamma_0, \hat{\alpha} : \kappa = \tau, \Gamma_1} \mathsf{InstSolve} \qquad \frac{\hat{\beta} \in \mathsf{unsolved}(\Gamma[\hat{\alpha} : \kappa][\hat{\beta} : \kappa])}{\Gamma[\hat{\alpha} : \kappa][\hat{\beta} : \kappa] \vdash \hat{\alpha} \coloneqq \hat{\beta} : \kappa \dashv \Gamma[\hat{\alpha} : \kappa][\hat{\beta} : \kappa = \hat{\alpha}]} \mathsf{InstReach}$$

$$\frac{\Gamma[\hat{\alpha}_2 : \star, \hat{\alpha}_1 : \star, \hat{\alpha} : \star = \hat{\alpha}_1 \oplus \hat{\alpha}_2] \vdash \hat{\alpha}_1 \coloneqq \tau_1 : \star \dashv \Theta \qquad \Theta \vdash \hat{\alpha}_2 \coloneqq [\Theta]\tau_2 : \star \dashv \Delta}{\Gamma[\hat{\alpha} : \star] \vdash \hat{\alpha} \coloneqq \tau_1 \oplus \tau_2 : \star \dashv \Delta} \mathsf{InstBin}$$

$$\frac{}{\Gamma[\hat{\alpha} : \mathbb{N}] \vdash \hat{\alpha} \coloneqq \mathsf{zero} : \mathbb{N} \dashv \Gamma[\hat{\alpha} : \mathbb{N} = \mathsf{zero}]} \mathsf{InstZero} \qquad \frac{\Gamma[\hat{\alpha}_1 : \mathbb{N}, \hat{\alpha} : \mathbb{N} = \mathsf{succ}(\hat{\alpha}_1)] \vdash \hat{\alpha}_1 \coloneqq t_1 : \mathbb{N} \dashv \Delta}{\Gamma[\hat{\alpha} : \mathbb{N}] \vdash \hat{\alpha} \coloneqq \mathsf{succ}(t_1) : \mathbb{N} \dashv \Delta} \mathsf{InstSucc}$$

Figure 24. Instantiation

$\boxed{\Gamma \vdash \Pi :: \vec{A} \Leftarrow C\ p\ \dashv \Delta}$ Under context Γ,
check branches Π with patterns of type $\vec{A}$ and bodies of type C

$$\frac{}{\Gamma \vdash \cdot :: \vec{A} \Leftarrow C\ p\ \dashv \Gamma}\ \mathsf{MatchEmpty} \qquad \frac{\Gamma \vdash \pi :: \vec{A} \Leftarrow C\ p\ \dashv \Theta \qquad \Theta \vdash \Pi' :: \vec{A} \Leftarrow C\ p\ \dashv \Delta}{\Gamma \vdash \pi \mid \Pi' :: \vec{A} \Leftarrow C\ p\ \dashv \Delta}\ \mathsf{MatchSeq}$$

$$\frac{\Gamma \vdash e \Leftarrow C\ p\ \dashv \Delta}{\Gamma \vdash (\cdot \Rightarrow e) :: \cdot \Leftarrow C\ p\ \dashv \Delta}\ \mathsf{MatchBase} \qquad \frac{\Gamma \vdash \vec{\rho} \Rightarrow e :: \vec{A} \Leftarrow C\ p\ \dashv \Delta}{\Gamma \vdash (), \vec{\rho} \Rightarrow e :: 1, \vec{A} \Leftarrow C\ p\ \dashv \Delta}\ \mathsf{MatchUnit}$$

$$\frac{\Gamma, \alpha : \kappa \vdash \vec{\rho} \Rightarrow e :: A, \vec{A} \Leftarrow C\ p\ \dashv \Delta, \alpha : \kappa, \Theta}{\Gamma \vdash \vec{\rho} \Rightarrow e :: (\exists \alpha : \kappa.\ A), \vec{A} \Leftarrow C\ p\ \dashv \Delta}\ \mathsf{Match}\exists \qquad \frac{\Gamma \mathbin{/} P \vdash \vec{\rho} \Rightarrow e :: A, \vec{A} \Leftarrow C\ p\ \dashv \Delta}{\Gamma \vdash \vec{\rho} \Rightarrow e :: A \wedge P, \vec{A} \Leftarrow C\ p\ \dashv \Delta}\ \mathsf{Match}\wedge$$

$$\frac{\Gamma \vdash \rho_1, \rho_2, \vec{\rho} \Rightarrow e :: A_1, A_2, \vec{A} \Leftarrow C\ p\ \dashv \Delta}{\Gamma \vdash \langle \rho_1, \rho_2 \rangle, \vec{\rho} \Rightarrow e :: A_1 \times A_2, \vec{A} \Leftarrow C\ p\ \dashv \Delta}\ \mathsf{Match}\times \qquad \frac{\Gamma \vdash \rho, \vec{\rho} \Rightarrow e :: A_k, \vec{A} \Leftarrow C\ p\ \dashv \Delta}{\Gamma \vdash (\mathsf{inj}_k\ \rho), \vec{\rho} \Rightarrow e :: A_1 + A_2, \vec{A} \Leftarrow C\ p\ \dashv \Delta}\ \mathsf{Match}+_k$$

$$\frac{A \text{ not headed by } \wedge \text{ or } \exists \qquad \Gamma, z : A\,! \vdash \vec{\rho} \Rightarrow e' :: \vec{A} \Leftarrow C\ p\ \dashv \Delta, z : A\,!, \Delta'}{\Gamma \vdash z, \vec{\rho} \Rightarrow e :: A, \vec{A} \Leftarrow C\ p\ \dashv \Delta}\ \mathsf{MatchNeg}$$

$$\frac{A \text{ not headed by } \wedge \text{ or } \exists \qquad \Gamma \vdash \vec{\rho} \Rightarrow e :: \vec{A} \Leftarrow C\ p\ \dashv \Delta}{\Gamma \vdash \_, \vec{\rho} \Rightarrow e :: A, \vec{A} \Leftarrow C\ p\ \dashv \Delta}\ \mathsf{MatchWild}$$

$$\frac{\Gamma \mathbin{/} (t = \mathsf{zero}) \vdash \vec{\rho} \Rightarrow e :: \vec{A} \Leftarrow C\ p\ \dashv \Delta}{\Gamma \vdash [], \vec{\rho} \Rightarrow e :: (\mathsf{Vec}\ t\ A), \vec{A} \Leftarrow C\ p\ \dashv \Delta}\ \mathsf{MatchNil}$$

$$\frac{\Gamma, \alpha : \mathbb{N} \mathbin{/} (t = \mathsf{succ}(\alpha)) \vdash \rho_1, \rho_2, \vec{\rho} \Rightarrow e :: A, (\mathsf{Vec}\ \alpha\ A), \vec{A} \Leftarrow C\ p\ \dashv \Delta, \alpha : \mathbb{N}, \Theta}{\Gamma \vdash (\rho_1 :: \rho_2), \vec{\rho} \Rightarrow e :: (\mathsf{Vec}\ t\ A), \vec{A} \Leftarrow C\ p\ \dashv \Delta}\ \mathsf{MatchCons}$$

$\boxed{\Gamma \mathbin{/} P \vdash \Pi :: \vec{A} \Leftarrow C\ p\ \dashv \Delta}$ Under context Γ, incorporate proposition P while checking branches Π
with patterns of type $\vec{A}$ and bodies of type C

$$\frac{\Gamma \mathbin{/} \sigma \doteq \tau : \kappa\ \dashv \bot}{\Gamma \mathbin{/} \sigma = \tau \vdash \vec{\rho} \Rightarrow e :: \vec{A} \Leftarrow C\ p\ \dashv \Gamma}\ \mathsf{Match}\bot \qquad \frac{\Gamma, \blacktriangleright_P \mathbin{/} \sigma \doteq \tau : \kappa\ \dashv \Theta \qquad \Theta \vdash \vec{\rho} \Rightarrow e :: \vec{A} \Leftarrow C\ p\ \dashv \Delta, \blacktriangleright_P, \Delta'}{\Gamma \mathbin{/} \sigma = \tau \vdash \vec{\rho} \Rightarrow e :: \vec{A} \Leftarrow C\ p\ \dashv \Delta}\ \mathsf{MatchUnify}$$

Figure 25. Algorithmic pattern matching

$\boxed{\Gamma \vdash \Pi\ \mathit{covers}\ \vec{A}}$ Under context Γ, patterns Π cover the types $\vec{A}$

$\boxed{\Gamma \mathbin{/} P \vdash \Pi\ \mathit{covers}\ \vec{A}}$ Under context Γ, patterns Π cover the types $\vec{A}$ assuming P

$$\frac{}{\Gamma \vdash (\cdot \Rightarrow e_1) \mid \Pi\ \mathit{covers}\ \cdot}\ \mathsf{CoversEmpty} \qquad \frac{\Pi \overset{\mathsf{var}}{\rightsquigarrow} \Pi' \qquad \Gamma \vdash \Pi'\ \mathit{covers}\ \vec{A}}{\Gamma \vdash \Pi\ \mathit{covers}\ A, \vec{A}}\ \mathsf{CoversVar} \qquad \frac{\Pi \overset{1}{\rightsquigarrow} \Pi' \qquad \Gamma \vdash \Pi'\ \mathit{covers}\ \vec{A}}{\Gamma \vdash \Pi\ \mathit{covers}\ 1, \vec{A}}\ \mathsf{Covers1}$$

$$\frac{\Pi \overset{\times}{\rightsquigarrow} \Pi' \qquad \Gamma \vdash \Pi'\ \mathit{covers}\ A_1, A_2, \vec{A}}{\Gamma \vdash \Pi\ \mathit{covers}\ (A_1 \times A_2), \vec{A}}\ \mathsf{Covers}\times \qquad \frac{\Pi \overset{+}{\rightsquigarrow} \Pi_L \parallel \Pi_R \qquad \Gamma \vdash \Pi_L\ \mathit{covers}\ A_1, \vec{A} \qquad \Gamma \vdash \Pi_R\ \mathit{covers}\ A_2, \vec{A}}{\Gamma \vdash \Pi\ \mathit{covers}\ (A_1 + A_2), \vec{A}}\ \mathsf{Covers}+$$

$$\frac{\Gamma, \alpha : \kappa \vdash \Pi\ \mathit{covers}\ \vec{A}}{\Gamma \vdash \Pi\ \mathit{covers}\ (\exists \alpha : \kappa.\ A), \vec{A}}\ \mathsf{Covers}\exists \qquad \frac{\Gamma \mathbin{/} t_1 = t_2 \vdash \Pi\ \mathit{covers}\ A_0, \vec{A}}{\Gamma \vdash \Pi\ \mathit{covers}\ \big(A_0 \wedge (t_1 = t_2)\big), \vec{A}}\ \mathsf{Covers}\wedge$$

$$\frac{\Pi \overset{\mathsf{Vec}}{\rightsquigarrow} \Pi_{[]} \parallel \Pi_{::} \qquad \Gamma \mathbin{/} t = \mathsf{zero} \vdash \Pi_{[]}\ \mathit{covers}\ \vec{A} \qquad \Gamma, n : \mathbb{N} \mathbin{/} t = \mathsf{succ}(n) \vdash \Pi_{::}\ \mathit{covers}\ (A, \mathsf{Vec}\ n\ A, \vec{A})}{\Gamma \vdash \Pi\ \mathit{covers}\ \mathsf{Vec}\ t\ A, \vec{A}}\ \mathsf{CoversVec}$$

$$\frac{\Gamma \mathbin{/} [\Gamma]t_1 \doteq [\Gamma]t_2 : \kappa\ \dashv \Delta \qquad \Delta \vdash [\Delta]\Pi\ \mathit{covers}\ [\Delta]\vec{A}}{\Gamma \mathbin{/} t_1 = t_2 \vdash \Pi\ \mathit{covers}\ \vec{A}}\ \mathsf{CoversEq} \qquad \frac{\Gamma \mathbin{/} [\Gamma]t_1 \doteq [\Gamma]t_2 : \kappa\ \dashv \bot}{\Gamma \mathbin{/} t_1 = t_2 \vdash \Pi\ \mathit{covers}\ \vec{A}}\ \mathsf{CoversEqBot}$$

Figure 26. Algorithmic match coverage

E. Theorems and Selected Lemmas

E.1 Decidability and determinacy

Theorem 1 (Decidability of Subtyping)**.**
Given a context Γ *and types* A, B *such that* $\Gamma \vdash A$ *type and* $\Gamma \vdash B$ *type and* $[\Gamma]A = A$ *and* $[\Gamma]B = B$, *it is decidable whether there exists* Δ *such that* $\Gamma \vdash A <:^{\pm} B \dashv \Delta$.

Theorem 2 (Decidability of Coverage)**.**
Given a context Γ, *branches* Π *and types* $\vec{A}$, *it is decidable whether* $\Gamma \vdash \Pi$ *covers* $\vec{A}$ *is derivable.*

Theorem 3 (Decidability of Typing)**.**

(i) Synthesis: *Given a context* Γ, *a principality* p, *and a term* e,
it is decidable whether there exist a type A *and a context* Δ *such that*
$\Gamma \vdash e \Rightarrow A\ p \dashv \Delta$.

(ii) Spines: *Given a context* Γ, *a spine* s, *a principality* p, *and a type* A *such that* $\Gamma \vdash A$ *type,*
it is decidable whether there exist a type B, *a principality* q *and a context* Δ *such that*
$\Gamma \vdash s : A\ p \gg B\ q \dashv \Delta$.

(iii) Checking: *Given a context* Γ, *a principality* p, *a term* e, *and a type* B *such that* $\Gamma \vdash B$ *type,*
it is decidable whether there is a context Δ *such that*
$\Gamma \vdash e \Leftarrow B\ p \dashv \Delta$.

(iv) Matching: *Given a context* Γ, *branches* Π, *a list of types* $\vec{A}$, *a type* C, *and a principality* p, *it is decidable whether there exists* Δ *such that* $\Gamma \vdash \Pi :: \vec{A} \Leftarrow C\ p \dashv \Delta$.
Also, if given a proposition P *as well, it is decidable whether there exists* Δ *such that* $\Gamma \,/\, P \vdash \Pi :: \vec{A} \Leftarrow C\ p \dashv \Delta$.

Theorem 4 (Determinacy of Subtyping)**.**

(1) Subtyping: *Given* Γ, e, A, B *such that* $\mathcal{D}_1 :: \Gamma \vdash A <:^{\pm} B \dashv \Delta_1$ *and* $\mathcal{D}_2 :: \Gamma \vdash A <:^{\pm} B \dashv \Delta_2$,
it is the case that $\Delta_1 = \Delta_2$.

Theorem 5 (Determinacy of Typing)**.**

(1) Checking: *Given* Γ, e, A, p *such that* $\mathcal{D}_1 :: \Gamma \vdash e \Leftarrow A\ p \dashv \Delta_1$ *and* $\mathcal{D}_2 :: \Gamma \vdash e \Leftarrow A\ p \dashv \Delta_2$,
it is the case that $\Delta_1 = \Delta_2$.

(2) Synthesis: *Given* Γ, e *such that* $\mathcal{D}_1 :: \Gamma \vdash e \Rightarrow B_1\ p_1 \dashv \Delta_1$ *and* $\mathcal{D}_2 :: \Gamma \vdash e \Rightarrow B_2\ p_2 \dashv \Delta_2$,
it is the case that $B_1 = B_2$ *and* $p_1 = p_2$ *and* $\Delta_1 = \Delta_2$.

(3) Spine judgments:
Given Γ, e, A, p *such that* $\mathcal{D}_1 :: \Gamma \vdash e : A\ p \gg C_1\ q_1 \dashv \Delta_1$ *and* $\mathcal{D}_2 :: \Gamma \vdash e : A\ p \gg C_2\ q_2 \dashv \Delta_2$,
it is the case that $C_1 = C_2$ *and* $q_1 = q_2$ *and* $\Delta_1 = \Delta_2$.
The same applies for derivations of the principality-recovering judgments $\Gamma \vdash e : A\ p \gg C_k\ \lceil q_k \rceil \dashv \Delta_k$.

(4) Match judgments:
Given Γ, Π, $\vec{A}$, p, C *such that* $\mathcal{D}_1 :: \Gamma \vdash \Pi :: \vec{A} \Leftarrow C\ p \dashv \Delta_1$ *and* $\mathcal{D}_2 :: \Gamma \vdash \Pi :: \vec{A} \Leftarrow C\ p \dashv \Delta_2$,
it is the case that $\Delta_1 = \Delta_2$.
Given Γ, P, Π, $\vec{A}$, p, C
such that $\mathcal{D}_1 :: \Gamma \,/\, P \vdash \Pi :: \vec{A} \Leftarrow C\ p \dashv \Delta_1$ *and* $\mathcal{D}_2 :: \Gamma \,/\, P \vdash \Pi :: \vec{A} \Leftarrow C\ p \dashv \Delta_2$,
it is the case that $\Delta_1 = \Delta_2$.

E.2 Soundness

For several auxiliary judgment forms, soundness is a matter of showing that, given two algorithmic terms, their declarative versions are equal. For example, for the instantiation judgment we have:

Lemma (Soundness of Instantiation)**.**
If $\Gamma \vdash \hat{\alpha} := \tau : \kappa \dashv \Delta$ *and* $\hat{\alpha} \notin \mathrm{FV}([\Gamma]\tau)$ *and* $[\Gamma]\tau = \tau$ *and* $\Delta \longrightarrow \Omega$ *then* $[\Omega]\hat{\alpha} = [\Omega]\tau$.

We have similar lemmas for term equality ($\Gamma \vdash \sigma \doteq t : \kappa \dashv \Delta$), propositional equivalence ($\Gamma \vdash P \equiv Q \dashv \Delta$) and type equivalence ($\Gamma \vdash A \equiv B \dashv \Delta$).

Our eliminating judgments incorporate assumptions into the context Γ. We show that the algorithmic rules for these judgments just append equations over universal variables:

Lemma (Soundness of Equality Elimination)**.** *If* $[\Gamma]\sigma = \sigma$ *and* $[\Gamma]t = t$ *and* $\Gamma \vdash \sigma : \kappa$ *and* $\Gamma \vdash t : \kappa$ *and* $\mathsf{FEV}(\sigma) \cup \mathsf{FEV}(t) = \emptyset$, *then:*

(1) *If* $\Gamma \,/\, \sigma \doteq t : \kappa \dashv \Delta$
then $\Delta = (\Gamma, \Theta)$ *where* $\Theta = (\alpha_1 = t_1, \ldots, \alpha_n = t_n)$ *and*
for all Ω *such that* $\Gamma \longrightarrow \Omega$ *and all* t' *s.t.* $\Omega \vdash t' : \kappa'$
we have $[\Omega, \Theta]t' = [\theta][\Omega]t'$ *where* $\theta = \mathsf{mgu}(\sigma, t)$.

(2) *If* $\Gamma \,/\, \sigma \doteq t : \kappa \dashv \bot$ *then no most general unifier exists.*

The last lemmas for soundness move directly from an algorithmic judgment to the corresponding declarative judgment.

Lemma (Soundness of Checkprop)**.**
If $\Gamma \vdash P$ *true* $\dashv \Delta$ *and* $\Delta \longrightarrow \Omega$ *then* $\Psi \vdash [\Omega]P$ *true.*

Lemma (Soundness of Algorithmic Subtyping)**.** *If* $[\Gamma]A = A$ *and* $[\Gamma]B = B$ *and* $\Gamma \vdash A$ *type and* $\Gamma \vdash B$ *type and* $\Delta \longrightarrow \Omega$ *and* $\Gamma \vdash A <:^{\pm} B \dashv \Delta$ *then* $[\Omega]\Delta \vdash [\Omega]A \leq^{\pm} [\Omega]B$.

Lemma (Soundness of Match Coverage)**.**

1. *If* $\Gamma \vdash \Pi$ *covers* $\vec{A}$ *and* $\Gamma \longrightarrow \Omega$ *and* $\Gamma \vdash \vec{A}$ **!** *types and* $[\Gamma]\vec{A} = \vec{A}$ *then* $[\Omega]\Gamma \vdash \Pi$ *covers* $\vec{A}$.
2. *If* $\Gamma \,/\, P \vdash \Pi$ *covers* $\vec{A}$ *and* $\Gamma \longrightarrow \Omega$ *and* $\Gamma \vdash \vec{A}$ **!** *types and* $[\Gamma]\vec{A} = \vec{A}$ *and* $[\Gamma]P = P$ *then* $[\Omega]\Gamma \,/\, P \vdash \Pi$ *covers* $\vec{A}$.

Theorem 6 (Soundness of Algorithmic Subtyping)**.**
If $[\Gamma]A = A$ *and* $[\Gamma]B = B$ *and* $\Gamma \vdash A$ *type and* $\Gamma \vdash B$ *type and* $\Delta \longrightarrow \Omega$ *and* $\Gamma \vdash A <:^{\pm} B \dashv \Delta$ *then* $[\Omega]\Delta \vdash [\Omega]A \leq^{\pm} [\Omega]B$.

Theorem 7 (Soundness of Match Coverage)**.**
If $\Gamma \vdash \Pi$ *covers* $\vec{A}$ *and* $\Gamma \longrightarrow \Omega$ *and* $\Gamma \vdash \vec{A}$ **!** *types and* $[\Gamma]\vec{A} = \vec{A}$
then $[\Omega]\Gamma \vdash \Pi$ *covers* $\vec{A}$.

Theorem 8 (Soundness of Algorithmic Typing)**.**
Given $\Delta \longrightarrow \Omega$*:*

(i) *If* $\Gamma \vdash e \Leftarrow A\ p \dashv \Delta$ *and* $\Gamma \vdash A\ p$ *type then* $[\Omega]\Delta \vdash [\Omega]e \Leftarrow [\Omega]A\ p$.
(ii) *If* $\Gamma \vdash e \Rightarrow A\ p \dashv \Delta$ *then* $[\Omega]\Delta \vdash [\Omega]e \Rightarrow [\Omega]A\ p$.
(iii) *If* $\Gamma \vdash s : A\ p \gg B\ q \dashv \Delta$ *and* $\Gamma \vdash A\ p$ *type then* $[\Omega]\Delta \vdash [\Omega]s : [\Omega]A\ p \gg [\Omega]B\ q$.
(iv) *If* $\Gamma \vdash s : A\ p \gg B\ \lceil q \rceil \dashv \Delta$ *and* $\Gamma \vdash A\ p$ *type then* $[\Omega]\Delta \vdash [\Omega]s : [\Omega]A\ p \gg [\Omega]B\ \lceil q \rceil$.
(v) *If* $\Gamma \vdash \Pi :: \vec{A} \Leftarrow C\ p \dashv \Delta$ *and* $\Gamma \vdash \vec{A}$ **!** *types and* $[\Gamma]\vec{A} = \vec{A}$ *and* $\Gamma \vdash C\ p$ *type*
then $[\Omega]\Delta \vdash [\Omega]\Pi :: [\Omega]\vec{A} \Leftarrow [\Omega]C\ p$.
(vi) *If* $\Gamma \,/\, P \vdash \Pi :: \vec{A} \Leftarrow C\ p \dashv \Delta$ *and* $\Gamma \vdash P$ *prop and* $\mathsf{FEV}(P) = \emptyset$ *and* $[\Gamma]P = P$
and $\Gamma \vdash \vec{A}$ **!** *types and* $\Gamma \vdash C\ p$ *type*
then $[\Omega]\Delta \,/\, [\Omega]P \vdash [\Omega]\Pi :: [\Omega]\vec{A} \Leftarrow [\Omega]C\ p$.

E.3 Completeness

Theorem 9 (Completeness of Subtyping)**.**
If $\Gamma \longrightarrow \Omega$ *and* $\mathsf{dom}(\Gamma) = \mathsf{dom}(\Omega)$ *and* $\Gamma \vdash A$ *type and* $\Gamma \vdash B$ *type*
and $[\Omega]\Gamma \vdash [\Omega]A \leq^{\pm} [\Omega]B$
then there exist Δ *and* Ω' *such that* $\Delta \longrightarrow \Omega'$
and $\mathsf{dom}(\Delta) = \mathsf{dom}(\Omega')$
and $\Omega \longrightarrow \Omega'$
and $\Gamma \vdash [\Gamma]A <:^{\pm} [\Gamma]B \dashv \Delta$.

Theorem 10 (Completeness of Match Coverage)**.**

1. *If* $[\Omega]\Gamma \vdash [\Omega]\Pi$ *covers* $[\Omega]\vec{A}$ *and* $\Gamma \longrightarrow \Omega$ *and* $\Gamma \vdash \vec{A}$ **!** *types and* $[\Gamma]\vec{A} = \vec{A}$
then $\Gamma \vdash \Pi$ *covers* $\vec{A}$.
2. *If* $[\Omega]\Gamma \,/\, [\Omega]P \vdash [\Omega]\Pi$ *covers* $[\Omega]\vec{A}$ *and* $\Gamma \longrightarrow \Omega$ *and* $\Gamma \vdash \vec{A}$ **!** *types and* $[\Gamma]\vec{A} = \vec{A}$ *and* $[\Gamma]P = P$
then $\Gamma \,/\, P \vdash \Pi$ *covers* $\vec{A}$.

Theorem 11 (Completeness of Algorithmic Typing)**.** *Given* $\Gamma \longrightarrow \Omega$ *such that* $\mathsf{dom}(\Gamma) = \mathsf{dom}(\Omega)$*:*

(i) *If* $\Gamma \vdash A\ p$ *type and* $[\Omega]\Gamma \vdash [\Omega]e \Leftarrow [\Omega]A\ p$ *and* $p' \sqsubseteq p$
then there exist Δ *and* Ω'
such that $\Delta \longrightarrow \Omega'$ *and* $\mathsf{dom}(\Delta) = \mathsf{dom}(\Omega')$ *and* $\Omega \longrightarrow \Omega'$
and $\Gamma \vdash e \Leftarrow [\Gamma]A\ p' \dashv \Delta$.
(ii) *If* $\Gamma \vdash A\ p$ *type and* $[\Omega]\Gamma \vdash [\Omega]e \Rightarrow A\ p$
then there exist Δ, Ω', A', *and* $p' \sqsubseteq p$
such that $\Delta \longrightarrow \Omega'$ *and* $\mathsf{dom}(\Delta) = \mathsf{dom}(\Omega')$ *and* $\Omega \longrightarrow \Omega'$
and $\Gamma \vdash e \Rightarrow A'\ p' \dashv \Delta$ *and* $A' = [\Delta]A'$ *and* $A = [\Omega']A'$.
(iii) *If* $\Gamma \vdash A\ p$ *type and* $[\Omega]\Gamma \vdash [\Omega]s : [\Omega]A\ p \gg B\ q$ *and* $p' \sqsubseteq p$
then there exist Δ, Ω', B' *and* $q' \sqsubseteq q$
such that $\Delta \longrightarrow \Omega'$ *and* $\mathsf{dom}(\Delta) = \mathsf{dom}(\Omega')$ *and* $\Omega \longrightarrow \Omega'$
and $\Gamma \vdash s : [\Gamma]A\ p' \gg B'\ q' \dashv \Delta$ *and* $B' = [\Delta]B'$ *and* $B = [\Omega']B'$.
(iv) *If* $\Gamma \vdash A\ p$ *type and* $[\Omega]\Gamma \vdash [\Omega]s : [\Omega]A\ p \gg B\ \lceil q \rceil$ *and* $p' \sqsubseteq p$
then there exist Δ, Ω', B', *and* $q' \sqsubseteq q$
such that $\Delta \longrightarrow \Omega'$ *and* $\mathsf{dom}(\Delta) = \mathsf{dom}(\Omega')$ *and* $\Omega \longrightarrow \Omega'$
and $\Gamma \vdash s : [\Gamma]A\ p' \gg B'\ \lceil q' \rceil \dashv \Delta$ *and* $B' = [\Delta]B'$ *and* $B = [\Omega']B'$.

(v) *If* $\Gamma \vdash \vec{A}$ **!** *types and* $\Gamma \vdash C\ p$ *type and* $[\Omega]\Gamma \vdash [\Omega]\Pi :: [\Omega]\vec{A} \Leftarrow [\Omega]C\ p$ *and* $p' \sqsubseteq p$
then there exist Δ, Ω', *and* C
such that $\Delta \longrightarrow \Omega'$ *and* $\mathsf{dom}(\Delta) = \mathsf{dom}(\Omega')$ *and* $\Omega \longrightarrow \Omega'$
and $\Gamma \vdash \Pi :: [\Gamma]\vec{A} \Leftarrow [\Gamma]C\ p' \dashv \Delta$.

(vi) *If* $\Gamma \vdash \vec{A}$ **!** *types and* $\Gamma \vdash P$ *prop and* $\mathsf{FEV}(P) = \emptyset$ *and* $\Gamma \vdash C\ p$ *type*
and $[\Omega]\Gamma\ /\ [\Omega]P \vdash [\Omega]\Pi :: [\Omega]\vec{A} \Leftarrow [\Omega]C\ p$
and $p' \sqsubseteq p$
then there exist Δ, Ω', *and* C
such that $\Delta \longrightarrow \Omega'$ *and* $\mathsf{dom}(\Delta) = \mathsf{dom}(\Omega')$ *and* $\Omega \longrightarrow \Omega'$
and $\Gamma\ /\ [\Gamma]P \vdash \Pi :: [\Gamma]\vec{A} \Leftarrow [\Gamma]C\ p' \dashv \Delta$.

Sound and Complete Bidirectional Typechecking for Higher-Rank Polymorphism with Existentials and Indexed Types: (LICS 2016 submission 39): Full definitions, lemmas and proofs

Joshua Dunfield Neelakantan R. Krishnaswami

January 21, 2016

Contents

1 List of Judgments

For convenience, we list all the judgment forms:

Judgment	*Description*	*Location*
$\Psi \vdash t : \kappa$	Index term/monotype is well-formed	Figure ??
$\Psi \vdash P\ \mathit{prop}$	Proposition is well-formed	Figure ??
$\Psi \vdash A\ \mathit{type}$	Type is well-formed	Figure ??
$\Psi \vdash \vec{A}\ \mathit{types}$	Type vector is well-formed	Figure ??
$\Psi\ \mathit{ctx}$	Declarative context is well-formed	Figure ??
$\Psi \vdash A \leq^{\pm} B$	Declarative subtyping	Figure ??
$\Psi \vdash P\ \mathit{true}$	Declarative truth	Figure ??
$\Psi \vdash e \Leftarrow A\ p$	Declarative checking	Figure ??
$\Psi \vdash e \Rightarrow A\ p$	Declarative synthesis	Figure ??
$\Psi \vdash s : A\ p \gg C\ q$	Declarative spine typing	Figure ??
$\Psi \vdash s : A\ p \gg C\ \lceil q \rceil$	Declarative spine typing, recovering principality	Figure ??
$\Psi \vdash \Pi :: \vec{A} \Leftarrow C\ p$	Declarative pattern matching	Figure ??
$\Psi\ /\ P \vdash \Pi :: \vec{A} \Leftarrow C\ p$	Declarative proposition assumption	Figure ??
$\Psi \vdash \Pi\ \mathit{covers}\ \vec{A}$	Declarative match coverage	Figure ??
$\Gamma \vdash \tau : \kappa$	Index term/monotype is well-formed	Figure ??
$\Gamma \vdash P\ \mathit{prop}$	Proposition is well-formed	Figure ??
$\Gamma \vdash A\ \mathit{type}$	Polytype is well-formed	Figure ??
$\Gamma\ \mathit{ctx}$	Algorithmic context is well-formed	Figure ??
$[\Gamma]A$	Applying a context, as a substitution, to a type	Figure ??
$\Gamma \vdash P\ \mathit{true} \dashv \Delta$	Check proposition	Figure ??
$\Gamma\ /\ P \dashv \Delta^{\perp}$	Assume proposition	Figure ??
$\Gamma \vdash s \doteq t : \kappa \dashv \Delta$	Check equation	Figure ??
$s \# t$	Head constructors clash	Figure ??
$\Gamma\ /\ s \doteq t : \kappa \dashv \Delta^{\perp}$	Assume/eliminate equation	Figure ??
$\Gamma \vdash A <:^{\pm} B \dashv \Delta$	Algorithmic subtyping	Figure ??
$\Gamma\ /\ P \vdash A <: B \dashv \Delta$	Assume/eliminate proposition	Figure ??
$\Gamma \vdash P \equiv Q \dashv \Delta$	Equivalence of propositions	Figure ??
$\Gamma \vdash A \equiv B \dashv \Delta$	Equivalence of types	Figure ??
$\Gamma \vdash \hat{\alpha} := t : \kappa \dashv \Delta$	Instantiate	Figure ??
$e\ \mathit{chk\text{-}I}$	Checking intro form	Figure ??
$\Gamma \vdash e \Leftarrow A\ p \dashv \Delta$	Algorithmic checking	Figure ??
$\Gamma \vdash e \Rightarrow A\ p \dashv \Delta$	Algorithmic synthesis	Figure ??
$\Gamma \vdash s : A\ p \gg C\ q \dashv \Delta$	Algorithmic spine typing	Figure ??
$\Gamma \vdash s : A\ p \gg C\ \lceil q \rceil \dashv \Delta$	Algorithmic spine typing, recovering principality	Figure ??
$\Gamma \vdash \Pi :: \vec{A} \Leftarrow C\ p \dashv \Delta$	Algorithmic pattern matching	Figure ??
$\Gamma\ /\ P \vdash \Pi :: \vec{A} \Leftarrow C\ p \dashv \Delta$	Algorithmic pattern matching (assumption)	Figure ??
$\Gamma \vdash \Pi\ \mathit{covers}\ \vec{A}$	Algorithmic match coverage	Figure ??
$\Gamma \longrightarrow \Delta$	Context extension	Figure ??
$[\Omega]\Gamma$	Apply complete context	Figure ??

A Properties of the Declarative System

Lemma 1 (Declarative Well-foundedness). *Go to proof*
The inductive definition of the following judgments is well-founded:

(i) *synthesis* $\Psi \vdash e \Rightarrow B\ p$

(ii) *checking* $\Psi \vdash e \Leftarrow A\ p$

(iii) *checking, equality elimination* $\Psi\ /\ P \vdash e \Leftarrow C\ p$

(iv) *ordinary spine* $\Psi \vdash s : A\ p \gg B\ q$

(v) *recovery spine* $\Psi \vdash s : A\ p \gg B\ \lceil q \rceil$

(vi) *pattern matching* $\Psi \vdash \Pi :: \vec{A} \Leftarrow C\ p$

(vii) *pattern matching, equality elimination* $\Psi\ /\ P \vdash \Pi :: \vec{A} \Leftarrow C\ p$

Lemma 2 (Declarative Weakening). *Go to proof*

(i) *If* $\Psi_0, \Psi_1 \vdash t : \kappa$ *then* $\Psi_0, \Psi, \Psi_1 \vdash t : \kappa$.

(ii) *If* $\Psi_0, \Psi_1 \vdash P\ prop$ *then* $\Psi_0, \Psi, \Psi_1 \vdash P\ prop$.

(iii) *If* $\Psi_0, \Psi_1 \vdash P\ true$ *then* $\Psi_0, \Psi, \Psi_1 \vdash P\ true$.

(iv) *If* $\Psi_0, \Psi_1 \vdash A\ type$ *then* $\Psi_0, \Psi, \Psi_1 \vdash A\ type$.

Lemma 3 (Declarative Term Substitution). *Go to proof*
Suppose $\Psi \vdash t : \kappa$. *Then:*

1. *If* $\Psi_0, \alpha : \kappa, \Psi_1 \vdash t' : \kappa$ *then* $\Psi_0, [t/\alpha]\Psi_1 \vdash [t/\alpha]t' : \kappa$.
2. *If* $\Psi_0, \alpha : \kappa, \Psi_1 \vdash P\ prop$ *then* $\Psi_0, [t/\alpha]\Psi_1 \vdash [t/\alpha]P\ prop$.
3. *If* $\Psi_0, \alpha : \kappa, \Psi_1 \vdash A\ type$ *then* $\Psi_0, [t/\alpha]\Psi_1 \vdash [t/\alpha]A\ type$.
4. *If* $\Psi_0, \alpha : \kappa, \Psi_1 \vdash A \leq^{\pm} B$ *then* $\Psi_0, [t/\alpha]\Psi_1 \vdash [t/\alpha]A \leq^{\pm} [t/\alpha]B$.
5. *If* $\Psi_0, \alpha : \kappa, \Psi_1 \vdash P\ true$ *then* $\Psi_0, [t/\alpha]\Psi_1 \vdash [t/\alpha]P\ true$.

Lemma 4 (Reflexivity of Declarative Subtyping). *Go to proof*
Given $\Psi \vdash A\ type$, *we have that* $\Psi \vdash A \leq^{\pm} A$.

Lemma 5 (Subtyping Inversion). *Go to proof*

- *If* $\Psi \vdash \exists \alpha : \kappa.\ A \leq^{+} B$ *then* $\Psi, \alpha : \kappa \vdash A \leq^{+} B$.
- *If* $\Psi \vdash A \leq^{-} \forall \beta : \kappa.\ B$ *then* $\Psi, \beta : \kappa \vdash A \leq^{-} B$.

Lemma 6 (Subtyping Polarity Flip). *Go to proof*

- *If* $nonpos(A)$ *and* $nonpos(B)$ *and* $\Psi \vdash A \leq^{+} B$
  then $\Psi \vdash A \leq^{-} B$ *by a derivation of the same or smaller size.*
- *If* $nonneg(A)$ *and* $nonneg(B)$ *and* $\Psi \vdash A \leq^{-} B$
  then $\Psi \vdash A \leq^{+} B$ *by a derivation of the same or smaller size.*
- *If* $nonpos(A)$ *and* $nonneg(A)$ *and* $nonpos(B)$ *and* $nonneg(B)$ *and* $\Psi \vdash A \leq^{\pm} B$
  then $A = B$.

Lemma 7 (Transitivity of Declarative Subtyping). *Go to proof*
Given $\Psi \vdash A\ type$ *and* $\Psi \vdash B\ type$ *and* $\Psi \vdash C\ type$:

(i) *If* $\mathcal{D}_1 :: \Psi \vdash A \leq^{\pm} B$ *and* $\mathcal{D}_2 :: \Psi \vdash B \leq^{\pm} C$
then $\Psi \vdash A \leq^{\pm} C$.

Property 1. *We assume that all types mentioned in annotations in expressions have no free existential variables. By the grammar, it follows that all expressions have no free existential variables, that is,* $FEV(e) = \emptyset$.

B Substitution and Well-formedness Properties

Definition 1 (Softness). *A context Θ is* soft *iff it consists only of $\hat{\alpha} : \kappa$ and $\hat{\alpha} : \kappa = \tau$ declarations.*

Lemma 8 (Substitution—Well-formedness). *Go to proof*

(i) If $\Gamma \vdash A\ p$ type and $\Gamma \vdash \tau\ p$ type then $\Gamma \vdash [\tau/\alpha]A\ p$ type.

(ii) If $\Gamma \vdash P$ prop and $\Gamma \vdash \tau\ p$ type then $\Gamma \vdash [\tau/\alpha]P$ prop.
Moreover, if $p = !$ and $\mathsf{FEV}([\Gamma]P) = \emptyset$ then $\mathsf{FEV}([\Gamma][\tau/\alpha]P) = \emptyset$.

Lemma 9 (Uvar Preservation). *Go to proof*
If $\Delta \longrightarrow \Omega$ then:

(i) If $(\alpha : \kappa) \in \Omega$ then $(\alpha : \kappa) \in [\Omega]\Delta$.

(ii) If $(x : A\ p) \in \Omega$ then $(x : [\Omega]A\ p) \in [\Omega]\Delta$.

Lemma 10 (Sorting Implies Typing). *Go to proof If $\Gamma \vdash t : \star$ then $\Gamma \vdash t$ type.*

Lemma 11 (Right-Hand Substitution for Sorting). *Go to proof If $\Gamma \vdash t : \kappa$ then $\Gamma \vdash [\Gamma]t : \kappa$.*

Lemma 12 (Right-Hand Substitution for Propositions). *Go to proof If $\Gamma \vdash P$ prop then $\Gamma \vdash [\Gamma]P$ prop.*

Lemma 13 (Right-Hand Substitution for Typing). *Go to proof If $\Gamma \vdash A$ type then $\Gamma \vdash [\Gamma]A$ type.*

Lemma 14 (Substitution for Sorting). *Go to proof If $\Omega \vdash t : \kappa$ then $[\Omega]\Omega \vdash [\Omega]t : \kappa$.*

Lemma 15 (Substitution for Prop Well-Formedness). *Go to proof*
If $\Omega \vdash P$ prop then $[\Omega]\Omega \vdash [\Omega]P$ prop.

Lemma 16 (Substitution for Type Well-Formedness). *Go to proof If $\Omega \vdash A$ type then $[\Omega]\Omega \vdash [\Omega]A$ type.*

Lemma 17 (Substitution Stability). *Go to proof*
If (Ω, Ω_Z) is well-formed and Ω_Z is soft and $\Omega \vdash A$ type then $[\Omega]A = [\Omega, \Omega_Z]A$.

Lemma 18 (Equal Domains). *Go to proof*
If $\Omega_1 \vdash A$ type and $\mathsf{dom}(\Omega_1) = \mathsf{dom}(\Omega_2)$ then $\Omega_2 \vdash A$ type.

C Properties of Extension

Lemma 19 (Declaration Preservation). *Go to proof If $\Gamma \longrightarrow \Delta$ and u is declared in Γ, then u is declared in Δ.*

Lemma 20 (Declaration Order Preservation). *Go to proof If $\Gamma \longrightarrow \Delta$ and u is declared to the left of v in Γ, then u is declared to the left of v in Δ.*

Lemma 21 (Reverse Declaration Order Preservation). *Go to proof If $\Gamma \longrightarrow \Delta$ and u and v are both declared in Γ and u is declared to the left of v in Δ, then u is declared to the left of v in Γ.*

An older paper had a lemma

> "Substitution Extension Invariance"
> If $\Theta \vdash A$ type and $\Theta \longrightarrow \Gamma$ then $[\Gamma]A = [\Gamma]([\Theta]A)$ and $[\Gamma]A = [\Theta]([\Gamma]A)$.

For the second part, $[\Gamma]A = [\Theta]([\Gamma]A)$, use Lemma 29 (Substitution Monotonicity) (i) or (iii) instead. The first part $[\Gamma]A = [\Gamma][\Theta]A$ hasn't been proved in this system.

Lemma 22 (Extension Inversion). *Go to proof*

(i) If $\mathcal{D} :: \Gamma_0, \alpha : \kappa, \Gamma_1 \longrightarrow \Delta$
then there exist unique Δ_0 and Δ_1
such that $\Delta = (\Delta_0, \alpha : \kappa, \Delta_1)$ and $\mathcal{D}' :: \Gamma_0 \longrightarrow \Delta_0$ where $\mathcal{D}' < \mathcal{D}$.
Moreover, if Γ_1 is soft, then Δ_1 is soft.

(ii) *If* $\mathcal{D} :: \Gamma_0, \blacktriangleright_u, \Gamma_1 \longrightarrow \Delta$
then there exist unique Δ_0 *and* Δ_1
such that $\Delta = (\Delta_0, \blacktriangleright_u, \Delta_1)$ *and* $\mathcal{D}' :: \Gamma_0 \longrightarrow \Delta_0$ *where* $\mathcal{D}' < \mathcal{D}$.

Moreover, if Γ_1 *is soft, then* Δ_1 *is soft.*

Moreover, if $\mathsf{dom}(\Gamma_0, \blacktriangleright_u, \Gamma_1) = \mathsf{dom}(\Delta)$ *then* $\mathsf{dom}(\Gamma_0) = \mathsf{dom}(\Delta_0)$.

(iii) *If* $\mathcal{D} :: \Gamma_0, \alpha = \tau, \Gamma_1 \longrightarrow \Delta$
then there exist unique Δ_0, τ', *and* Δ_1
such that $\Delta = (\Delta_0, \alpha = \tau', \Delta_1)$ *and* $\mathcal{D}' :: \Gamma_0 \longrightarrow \Delta_0$ *and* $[\Delta_0]\tau = [\Delta_0]\tau'$ *where* $\mathcal{D}' < \mathcal{D}$.

(iv) *If* $\mathcal{D} :: \Gamma_0, \hat{\alpha} : \kappa = \tau, \Gamma_1 \longrightarrow \Delta$
then there exist unique Δ_0, τ', *and* Δ_1
such that $\Delta = (\Delta_0, \hat{\alpha} : \kappa = \tau', \Delta_1)$ *and* $\mathcal{D}' :: \Gamma_0 \longrightarrow \Delta_0$ *and* $[\Delta_0]\tau = [\Delta_0]\tau'$ *where* $\mathcal{D}' < \mathcal{D}$.

(v) *If* $\mathcal{D} :: \Gamma_0, x : A, \Gamma_1 \longrightarrow \Delta$
then there exist unique Δ_0, A', *and* Δ_1
such that $\Delta = (\Delta_0, x : A', \Delta_1)$ *and* $\mathcal{D}' :: \Gamma_0 \longrightarrow \Delta_0$ *and* $[\Delta_0]A = [\Delta_0]A'$ *where* $\mathcal{D}' < \mathcal{D}$.

Moreover, if Γ_1 *is soft, then* Δ_1 *is soft.*

Moreover, if $\mathsf{dom}(\Gamma_0, x : A, \Gamma_1) = \mathsf{dom}(\Delta)$ *then* $\mathsf{dom}(\Gamma_0) = \mathsf{dom}(\Delta_0)$.

(vi) *If* $\mathcal{D} :: \Gamma_0, \hat{\alpha} : \kappa, \Gamma_1 \longrightarrow \Delta$ *then either*

- *there exist unique* Δ_0, τ', *and* Δ_1
  such that $\Delta = (\Delta_0, \hat{\alpha} : \kappa = \tau', \Delta_1)$ *and* $\mathcal{D}' :: \Gamma_0 \longrightarrow \Delta_0$ *where* $\mathcal{D}' < \mathcal{D}$,
  or
- *there exist unique* Δ_0 *and* Δ_1
  such that $\Delta = (\Delta_0, \hat{\alpha} : \kappa, \Delta_1)$ *and* $\mathcal{D}' :: \Gamma_0 \longrightarrow \Delta_0$ *where* $\mathcal{D}' < \mathcal{D}$.

Lemma 23 (Deep Evar Introduction). *Go to proof*

(i) *If* Γ_0, Γ_1 *is well-formed and* $\hat{\alpha}$ *is not declared in* Γ_0, Γ_1 *then* $\Gamma_0, \Gamma_1 \longrightarrow \Gamma_0, \hat{\alpha} : \kappa, \Gamma_1$.

(ii) *If* $\Gamma_0, \hat{\alpha} : \kappa, \Gamma_1$ *is well-formed and* $\Gamma \vdash t : \kappa$ *then* $\Gamma_0, \hat{\alpha} : \kappa, \Gamma_1 \longrightarrow \Gamma_0, \hat{\alpha} : \kappa = t, \Gamma_1$.

(iii) *If* Γ_0, Γ_1 *is well-formed and* $\Gamma \vdash t : \kappa$ *then* $\Gamma_0, \Gamma_1 \longrightarrow \Gamma_0, \hat{\alpha} : \kappa = t, \Gamma_1$.

Lemma 24 (Soft Extension). *Go to proof*
If $\Gamma \longrightarrow \Delta$ *and* Γ, Θ *ctx and* Θ *is soft, then there exists* Ω *such that* $\mathsf{dom}(\Theta) = \mathsf{dom}(\Omega)$ *and* $\Gamma, \Theta \longrightarrow \Delta, \Omega$.

Definition 2 (Filling). *The* filling *of a context* $|\Gamma|$ *solves all unsolved variables:*

$$
\begin{array}{lcl}
|\cdot| & = & \cdot \\
|\Gamma, x : A| & = & |\Gamma|, x : A \\
|\Gamma, \alpha : \kappa| & = & |\Gamma|, \alpha : \kappa \\
|\Gamma, \alpha = t| & = & |\Gamma|, \alpha = t \\
|\Gamma, \hat{\alpha} : \kappa = t| & = & |\Gamma|, \hat{\alpha} : \kappa = t \\
|\Gamma, \blacktriangleright_{\hat{\alpha}}| & = & |\Gamma|, \blacktriangleright_{\hat{\alpha}} \\
|\Gamma, \hat{\alpha} : \star| & = & |\Gamma|, \hat{\alpha} : \star = 1 \\
|\Gamma, \hat{\alpha} : \mathbb{N}| & = & |\Gamma|, \hat{\alpha} : \mathbb{N} = \mathsf{zero}
\end{array}
$$

Lemma 25 (Filling Completes). *If* $\Gamma \longrightarrow \Omega$ *and* (Γ, Θ) *is well-formed, then* $\Gamma, \Theta \longrightarrow \Omega, |\Theta|$.

Proof. By induction on Θ, following the definition of $|-|$ and applying the rules for $\longrightarrow$. □

Lemma 26 (Parallel Admissibility). *Go to proof*
If $\Gamma_L \longrightarrow \Delta_L$ *and* $\Gamma_L, \Gamma_R \longrightarrow \Delta_L, \Delta_R$ *then:*

(i) $\Gamma_L, \hat{\alpha} : \kappa, \Gamma_R \longrightarrow \Delta_L, \hat{\alpha} : \kappa, \Delta_R$

(ii) *If* $\Delta_L \vdash \tau' : \kappa$ *then* $\Gamma_L, \hat{\alpha} : \kappa, \Gamma_R \longrightarrow \Delta_L, \hat{\alpha} : \kappa = \tau', \Delta_R$.

(iii) *If* $\Gamma_L \vdash \tau : \kappa$ *and* $\Delta_L \vdash \tau'$ *type and* $[\Delta_L]\tau = [\Delta_L]\tau'$, *then* $\Gamma_L, \hat{\alpha} : \kappa = \tau, \Gamma_R \longrightarrow \Delta_L, \hat{\alpha} : \kappa = \tau', \Delta_R$.

Lemma 27 (Parallel Extension Solution). Go to proof
If $\Gamma_L, \hat{\alpha} : \kappa, \Gamma_R \longrightarrow \Delta_L, \hat{\alpha} : \kappa = \tau', \Delta_R$ *and* $\Gamma_L \vdash \tau : \kappa$ *and* $[\Delta_L]\tau = [\Delta_L]\tau'$
then $\Gamma_L, \hat{\alpha} : \kappa = \tau, \Gamma_R \longrightarrow \Delta_L, \hat{\alpha} : \kappa = \tau', \Delta_R$.

Lemma 28 (Parallel Variable Update). Go to proof
If $\Gamma_L, \hat{\alpha} : \kappa, \Gamma_R \longrightarrow \Delta_L, \hat{\alpha} : \kappa = \tau_0, \Delta_R$ *and* $\Gamma_L \vdash \tau_1 : \kappa$ *and* $\Delta_L \vdash \tau_2 : \kappa$ *and* $[\Delta_L]\tau_0 = [\Delta_L]\tau_1 = [\Delta_L]\tau_2$
then $\Gamma_L, \hat{\alpha} : \kappa = \tau_1, \Gamma_R \longrightarrow \Delta_L, \hat{\alpha} : \kappa = \tau_2, \Delta_R$.

Lemma 29 (Substitution Monotonicity). Go to proof

(i) *If* $\Gamma \longrightarrow \Delta$ *and* $\Gamma \vdash t : \kappa$ *then* $[\Delta][\Gamma]t = [\Delta]t$.

(ii) *If* $\Gamma \longrightarrow \Delta$ *and* $\Gamma \vdash P$ *prop then* $[\Delta][\Gamma]P = [\Delta]P$.

(iii) *If* $\Gamma \longrightarrow \Delta$ *and* $\Gamma \vdash A$ *type then* $[\Delta][\Gamma]A = [\Delta]A$.

Lemma 30 (Substitution Invariance). Go to proof

(i) *If* $\Gamma \longrightarrow \Delta$ *and* $\Gamma \vdash t : \kappa$ *and* $\mathsf{FEV}([\Gamma]t) = \emptyset$ *then* $[\Delta][\Gamma]t = [\Gamma]t$.

(ii) *If* $\Gamma \longrightarrow \Delta$ *and* $\Gamma \vdash P$ *prop and* $\mathsf{FEV}([\Gamma]P) = \emptyset$ *then* $[\Delta][\Gamma]P = [\Gamma]P$.

(iii) *If* $\Gamma \longrightarrow \Delta$ *and* $\Gamma \vdash A$ *type and* $\mathsf{FEV}([\Gamma]A) = \emptyset$ *then* $[\Delta][\Gamma]A = [\Gamma]A$.

Definition 3 (Canonical Contexts). *A (complete) context* Ω *is* canonical *iff, for all* $(\hat{\alpha} : \kappa = t)$ *and* $(\alpha = t) \in \Omega$, *the solution* t *is ground (*$\mathsf{FEV}(t) = \emptyset$*).*

Lemma 31 (Split Extension). Go to proof
If $\Delta \longrightarrow \Omega$
and $\hat{\alpha} \in \mathsf{unsolved}(\Delta)$
and $\Omega = \Omega_1[\hat{\alpha} : \kappa = t_1]$
and Ω *is canonical (Definition 3)*
and $\Omega \vdash t_2 : \kappa$
then $\Delta \longrightarrow \Omega_1[\hat{\alpha} : \kappa = t_2]$.

C.1 Reflexivity and Transitivity

Lemma 32 (Extension Reflexivity). Go to proof
If Γ *ctx then* $\Gamma \longrightarrow \Gamma$.

Lemma 33 (Extension Transitivity). Go to proof
If $\mathcal{D} :: \Gamma \longrightarrow \Theta$ *and* $\mathcal{D}' :: \Theta \longrightarrow \Delta$ *then* $\Gamma \longrightarrow \Delta$.

C.2 Weakening

The "suffix weakening" lemmas take a judgment under Γ and produce a judgment under (Γ, Θ). They do *not* require $\Gamma \longrightarrow \Gamma, \Theta$.

Lemma 34 (Suffix Weakening). Go to proof *If* $\Gamma \vdash t : \kappa$ *then* $\Gamma, \Theta \vdash t : \kappa$.

Lemma 35 (Suffix Weakening). Go to proof *If* $\Gamma \vdash A$ *type then* $\Gamma, \Theta \vdash A$ *type.*

The following proposed lemma is false.

> "Extension Weakening (Truth)"
> If $\Gamma \vdash P$ *true* $\dashv \Delta$ and $\Gamma \longrightarrow \Gamma'$ then there exists Δ' such that $\Delta \longrightarrow \Delta'$ and $\Gamma' \vdash P$ *true* $\dashv \Delta'$.

Counterexample: Suppose $\hat{\alpha} \vdash \hat{\alpha} = 1$ *true* $\dashv \hat{\alpha} = 1$ and $\hat{\alpha} \longrightarrow (\hat{\alpha} = (1 \to 1))$. Then there does *not* exist such a Δ'.

Lemma 36 (Extension Weakening (Sorts)). Go to proof *If* $\Gamma \vdash t : \kappa$ *and* $\Gamma \longrightarrow \Delta$ *then* $\Delta \vdash t : \kappa$.

Lemma 37 (Extension Weakening (Props)). Go to proof *If* $\Gamma \vdash P$ *prop and* $\Gamma \longrightarrow \Delta$ *then* $\Delta \vdash P$ *prop.*

Lemma 38 (Extension Weakening (Types)). Go to proof *If* $\Gamma \vdash A$ *type and* $\Gamma \longrightarrow \Delta$ *then* $\Delta \vdash A$ *type.*

C.3 Principal Typing Properties

Lemma 39 (Principal Agreement). *Go to proof*

(i) If $\Gamma \vdash A\ !$ *type and* $\Gamma \longrightarrow \Delta$ *then* $[\Delta]A = [\Gamma]A$.

(ii) If $\Gamma \vdash P$ *prop and* $\mathsf{FEV}(P) = \emptyset$ *and* $\Gamma \longrightarrow \Delta$ *then* $[\Delta]P = [\Gamma]P$.

Lemma 40 (Right-Hand Subst. for Principal Typing). *Go to proof If* $\Gamma \vdash A\ p$ *type then* $\Gamma \vdash [\Gamma]A\ p$ *type.*

Lemma 41 (Extension Weakening for Principal Typing). *Go to proof If* $\Gamma \vdash A\ p$ *type and* $\Gamma \longrightarrow \Delta$ *then* $\Delta \vdash A\ p$ *type.*

Lemma 42 (Inversion of Principal Typing). *Go to proof*

(1) If $\Gamma \vdash (A \to B)\ p$ *type then* $\Gamma \vdash A\ p$ *type and* $\Gamma \vdash B\ p$ *type.*

(2) If $\Gamma \vdash (P \supset A)\ p$ *type then* $\Gamma \vdash P$ *prop and* $\Gamma \vdash A\ p$ *type.*

(3) If $\Gamma \vdash (A \wedge P)\ p$ *type then* $\Gamma \vdash P$ *prop and* $\Gamma \vdash A\ p$ *type.*

C.4 Instantiation Extends

Lemma 43 (Instantiation Extension). *Go to proof*
If $\Gamma \vdash \hat{\alpha} := \tau : \kappa \dashv \Delta$ *then* $\Gamma \longrightarrow \Delta$.

C.5 Equivalence Extends

Lemma 44 (Elimeq Extension). *Go to proof*
If $\Gamma\ /\ s \stackrel{\circ}{=} t : \kappa \dashv \Delta$ *then there exists* Θ *such that* $\Gamma, \Theta \longrightarrow \Delta$.

Lemma 45 (Elimprop Extension). *Go to proof*
If $\Gamma\ /\ P \dashv \Delta$ *then there exists* Θ *such that* $\Gamma, \Theta \longrightarrow \Delta$.

Lemma 46 (Checkeq Extension). *Go to proof*
If $\Gamma \vdash A \equiv B \dashv \Delta$ *then* $\Gamma \longrightarrow \Delta$.

Lemma 47 (Checkprop Extension). *Go to proof*
If $\Gamma \vdash P$ *true* $\dashv \Delta$ *then* $\Gamma \longrightarrow \Delta$.

Lemma 48 (Prop Equivalence Extension). *Go to proof*
If $\Gamma \vdash P \equiv Q \dashv \Delta$ *then* $\Gamma \longrightarrow \Delta$.

Lemma 49 (Equivalence Extension). *Go to proof*
If $\Gamma \vdash A \equiv B \dashv \Delta$ *then* $\Gamma \longrightarrow \Delta$.

C.6 Subtyping Extends

Lemma 50 (Subtyping Extension). *Go to proof If* $\Gamma \vdash A <:^{\mp} B \dashv \Delta$ *then* $\Gamma \longrightarrow \Delta$.

C.7 Typing Extends

Lemma 51 (Typing Extension). *Go to proof*
If $\Gamma \vdash e \Leftarrow A\ p \dashv \Delta$
or $\Gamma \vdash e \Rightarrow A\ p \dashv \Delta$
or $\Gamma \vdash s : A\ p \gg B\ q \dashv \Delta$
or $\Gamma \vdash \Pi :: \vec{A} \Leftarrow C\ p \dashv \Delta$
or $\Gamma\ /\ P \vdash \Pi :: \vec{A} \Leftarrow C\ p \dashv \Delta$
then $\Gamma \longrightarrow \Delta$.

C.8 Unfiled

Lemma 52 (Context Partitioning). *Go to proof*
If $\Delta, \blacktriangleright_{\hat{\alpha}}, \Theta \longrightarrow \Omega, \blacktriangleright_{\hat{\alpha}}, \Omega_Z$ *then there is a* Ψ *such that* $[\Omega, \blacktriangleright_{\hat{\alpha}}, \Omega_Z](\Delta, \blacktriangleright_{\hat{\alpha}}, \Theta) = [\Omega]\Delta, \Psi$.

Lemma 53 (Softness Goes Away).
If $\Delta, \Theta \longrightarrow \Omega, \Omega_Z$ *where* $\Delta \longrightarrow \Omega$ *and* Θ *is soft, then* $[\Omega, \Omega_Z](\Delta, \Theta) = [\Omega]\Delta$.

Proof. By induction on Θ, following the definition of $[\Omega]\Gamma$. □

Lemma 54 (Completing Stability). *Go to proof*
If $\Gamma \longrightarrow \Omega$ *then* $[\Omega]\Gamma = [\Omega]\Omega$.

Lemma 55 (Completing Completeness). *Go to proof*

(i) If $\Omega \longrightarrow \Omega'$ *and* $\Omega \vdash t : \kappa$ *then* $[\Omega]t = [\Omega']t$.

(ii) If $\Omega \longrightarrow \Omega'$ *and* $\Omega \vdash A$ *type then* $[\Omega]A = [\Omega']A$.

(iii) If $\Omega \longrightarrow \Omega'$ *then* $[\Omega]\Omega = [\Omega']\Omega'$.

Lemma 56 (Confluence of Completeness). *Go to proof*
If $\Delta_1 \longrightarrow \Omega$ *and* $\Delta_2 \longrightarrow \Omega$ *then* $[\Omega]\Delta_1 = [\Omega]\Delta_2$.

Lemma 57 (Multiple Confluence). *Go to proof*
If $\Delta \longrightarrow \Omega$ *and* $\Omega \longrightarrow \Omega'$ *and* $\Delta' \longrightarrow \Omega'$ *then* $[\Omega]\Delta = [\Omega']\Delta'$.

Lemma 58 (Bundled Substitution for Sorting). *If* $\Gamma \vdash t : \kappa$ *and* $\Gamma \longrightarrow \Omega$ *then* $[\Omega]\Gamma \vdash [\Omega]t : \kappa$.

Proof.

	$\Gamma \vdash t : \kappa$	Given
	$\Omega \vdash t : \kappa$	By Lemma 36 (Extension Weakening (Sorts))
	$[\Omega]\Omega \vdash [\Omega]t : \kappa$	By Lemma 14 (Substitution for Sorting)
	$\Omega \longrightarrow \Omega$	By Lemma 32 (Extension Reflexivity)
	$[\Omega]\Omega = [\Omega]\Gamma$	By Lemma 56 (Confluence of Completeness)
☞	$[\Omega]\Gamma \vdash [\Omega]t : \kappa$	By above equality

□

Lemma 59 (Canonical Completion). *Go to proof*
If $\Gamma \longrightarrow \Omega$
then there exists Ω_{canon} *such that* $\Gamma \longrightarrow \Omega_{canon}$ *and* $\Omega_{canon} \longrightarrow \Omega$ *and* $\mathsf{dom}(\Omega_{canon}) = \mathsf{dom}(\Gamma)$ *and, for all* $\hat{\alpha} : \kappa = \tau$ *and* $\alpha = \tau$ *in* Ω_{canon}, *we have* $\mathsf{FEV}(\tau) = \emptyset$.

The completion Ω_{canon} is "canonical" because (1) its domain exactly matches Γ and (2) its solutions τ have no evars. Note that it follows from Lemma 57 (Multiple Confluence) that $[\Omega_{canon}]\Gamma = [\Omega]\Gamma$.

Lemma 60 (Split Solutions). *Go to proof*
If $\Delta \longrightarrow \Omega$ *and* $\hat{\alpha} \in \mathsf{unsolved}(\Delta)$
then there exists $\Omega_1 = \Omega_1'[\hat{\alpha} : \kappa = t_1]$ *such that* $\Omega_1 \longrightarrow \Omega$ *and* $\Omega_2 = \Omega_1'[\hat{\alpha} : \kappa = t_2]$ *where* $\Delta \longrightarrow \Omega_2$ *and* $t_2 \neq t_1$ *and* Ω_2 *is canonical.*

D Internal Properties of the Declarative System

Lemma 61 (Interpolating With and Exists). *Go to proof*

(1) If $\mathcal{D} :: \Psi \vdash \Pi :: \vec{A} \Leftarrow C\ p$ *and* $\Psi \vdash P_0$ *true*
then $\mathcal{D}' :: \Psi \vdash \Pi :: \vec{A} \Leftarrow C \wedge P_0\ p$.

(2) If $\mathcal{D} :: \Psi \vdash \Pi :: \vec{A} \Leftarrow [\tau/\alpha]C_0\ p$ *and* $\Psi \vdash \tau : \kappa$
then $\mathcal{D}' :: \Psi \vdash \Pi :: \vec{A} \Leftarrow (\exists \alpha : \kappa.\ C_0)\ p$.

In both cases, the height of $\mathcal{D}'$ *is one greater than the height of* $\mathcal{D}$.
Moreover, similar properties hold for the eliminating judgment $\Psi\ /\ P \vdash \Pi :: \vec{A} \Leftarrow C\ p$.

Lemma 62 (Case Invertibility). *Go to proof*
If $\Psi \vdash \mathsf{case}(e_0, \Pi) \Leftarrow C\ p$
then $\Psi \vdash e_0 \Rightarrow A\ !$ *and* $\Psi \vdash \Pi :: A \Leftarrow C\ p$ *and* $\Psi \vdash \Pi$ *covers* A
where the height of each resulting derivation is strictly less than the height of the given derivation.

E Miscellaneous Properties of the Algorithmic System

Lemma 63 (Well-Formed Outputs of Typing). *Go to proof*

(Spines) *If* $\Gamma \vdash s : A\ q \gg C\ p \dashv \Delta$ *or* $\Gamma \vdash s : A\ q \gg C\ \lceil p \rceil \dashv \Delta$
and $\Gamma \vdash A\ q$ *type*
then $\Delta \vdash C\ p$ *type.*

(Synthesis) *If* $\Gamma \vdash e \Rightarrow A\ p \dashv \Delta$
then $A \vdash p$ *type.*

F Decidability of Instantiation

Lemma 64 (Left Unsolvedness Preservation). *Go to proof*
If $\underbrace{\Gamma_0, \hat\alpha, \Gamma_1}_{\Gamma} \vdash \hat\alpha := A : \kappa \dashv \Delta$ *and* $\hat\beta \in \mathsf{unsolved}(\Gamma_0)$ *then* $\hat\beta \in \mathsf{unsolved}(\Delta)$.

Lemma 65 (Left Free Variable Preservation). *Go to proof* *If* $\overbrace{\Gamma_0, \hat\alpha : \kappa, \Gamma_1}^{\Gamma} \vdash \hat\alpha := t : \kappa \dashv \Delta$ *and* $\Gamma \vdash s : \kappa'$ *and* $\hat\alpha \notin \mathrm{FV}([\Gamma]s)$ *and* $\hat\beta \in \mathsf{unsolved}(\Gamma_0)$ *and* $\hat\beta \notin \mathrm{FV}([\Gamma]s)$, *then* $\hat\beta \notin \mathrm{FV}([\Delta]s)$.

Lemma 66 (Instantiation Size Preservation). *Go to proof* *If* $\overbrace{\Gamma_0, \hat\alpha, \Gamma_1}^{\Gamma} \vdash \hat\alpha := \tau : \kappa \dashv \Delta$ *and* $\Gamma \vdash s : \kappa'$ *and* $\hat\alpha \notin \mathrm{FV}([\Gamma]s)$, *then* $|[\Gamma]s| = |[\Delta]s|$, *where* $|C|$ *is the plain size of the term* C.

Lemma 67 (Decidability of Instantiation). *Go to proof* *If* $\Gamma = \Gamma_0[\hat\alpha : \kappa']$ *and* $\Gamma \vdash t : \kappa$ *such that* $[\Gamma]t = t$ *and* $\hat\alpha \notin \mathrm{FV}(t)$, *then:*

(1) Either there exists Δ *such that* $\Gamma_0[\hat\alpha : \kappa'] \vdash \hat\alpha := t : \kappa \dashv \Delta$, *or not.*

G Separation

Definition 4 (Separation).
An algorithmic context Γ *is* separable *and written* $\Gamma_L * \Gamma_R$ *if (1)* $\Gamma = (\Gamma_L, \Gamma_R)$ *and (2) for all* $(\hat\alpha : \kappa = \tau) \in \Gamma_R$ *it is the case that* $\mathsf{FEV}(\tau) \subseteq \mathsf{dom}(\Gamma_R)$.

Any context Γ is separable into, at least, $\cdot * \Gamma$ and $\Gamma * \cdot$.

Definition 5 (Separation-Preserving Extension).
The separated context $\Gamma_L * \Gamma_R$ *extends to* $\Delta_L * \Gamma_R$, *written*

$$(\Gamma_L * \Gamma_R) \xrightarrow[*]{} (\Delta_L * \Delta_R)$$

if $(\Gamma_L, \Gamma_R) \longrightarrow (\Delta_L, \Delta_R)$ *and* $\mathsf{dom}(\Gamma_L) \subseteq \mathsf{dom}(\Delta_L)$ *and* $\mathsf{dom}(\Gamma_R) \subseteq \mathsf{dom}(\Delta_R)$.

Separation-preserving extension says that variables from one half don't "cross" into the other half. Thus, Δ_L may add existential variables to Γ_L, and Δ_R may add existential variables to Γ_R, but no variable from Γ_L ends up in Δ_R and no variable from Γ_R ends up in Δ_L.

It is necessary to write $(\Gamma_L * \Gamma_R) \xrightarrow[*]{} (\Delta_L * \Delta_R)$ rather than $(\Gamma_L * \Gamma_R) \longrightarrow (\Delta_L * \Delta_R)$, because only $\xrightarrow[*]{}$ includes the domain conditions. For example, $(\hat\alpha * \hat\beta) \longrightarrow (\hat\alpha, \hat\beta = \hat\alpha) * \cdot$, but the variable $\hat\beta$ has "crossed over" to the left of $*$ in the context $(\hat\alpha, \hat\beta = \hat\alpha) * \cdot$.

Lemma 68 (Transitivity of Separation). *Go to proof*
If $(\Gamma_L * \Gamma_R) \xrightarrow[*]{} (\Theta_L * \Theta_R)$ *and* $(\Theta_L * \Theta_R) \xrightarrow[*]{} (\Delta_L * \Delta_R)$
then $(\Gamma_L * \Gamma_R) \xrightarrow[*]{} (\Delta_L * \Delta_R)$.

Lemma 69 (Separation Truncation). *Go to proof*
If H *has the form* $\alpha : \kappa$ *or* $\blacktriangleright_{\hat{\alpha}}$ *or* $\blacktriangleright_P$ *or* $x : A\ p$
and $(\Gamma_L * (\Gamma_R, H)) \xrightarrow{-*} (\Delta_L * \Delta_R)$
then $(\Gamma_L * \Gamma_R) \xrightarrow{-*} (\Delta_L * \Delta_0)$ *where* $\Delta_R = (\Delta_0, H, \Theta)$.

Lemma 70 (Separation for Auxiliary Judgments). *Go to proof*

(i) *If* $\Gamma_L * \Gamma_R \vdash \sigma \doteq \tau : \kappa \dashv \Delta$
and $\mathsf{FEV}(\sigma) \cup \mathsf{FEV}(\tau) \subseteq \mathsf{dom}(\Gamma_R)$
then $\Delta = (\Delta_L * \Delta_R)$ *and* $(\Gamma_L * \Gamma_R) \xrightarrow{-*} (\Delta_L * \Delta_R)$.

(ii) *If* $\Gamma_L * \Gamma_R \vdash P\ \mathit{true} \dashv \Delta$
and $\mathsf{FEV}(P) \subseteq \mathsf{dom}(\Gamma_R)$
then $\Delta = (\Delta_L * \Delta_R)$ *and* $(\Gamma_L * \Gamma_R) \xrightarrow{-*} (\Delta_L * \Delta_R)$.

(iii) *If* $\Gamma_L * \Gamma_R \ / \ \sigma \doteq \tau : \kappa \dashv \Delta$
and $\mathsf{FEV}(\sigma) \cup \mathsf{FEV}(\tau) = \emptyset$
then $\Delta = (\Delta_L * (\Delta_R, \Theta))$ *and* $(\Gamma_L * (\Gamma_R, \Theta)) \xrightarrow{-*} (\Delta_L * \Delta_R)$.

(iv) *If* $\Gamma_L * \Gamma_R \ / \ P \dashv \Delta$
and $\mathsf{FEV}(P) = \emptyset$
then $\Delta = (\Delta_L * (\Delta_R, \Theta))$ *and* $(\Gamma_L * (\Gamma_R, \Theta)) \xrightarrow{-*} (\Delta_L * \Delta_R)$.

(v) *If* $\Gamma_L * \Gamma_R \vdash \hat{\alpha} := \tau : \kappa \dashv \Delta$
and $(\mathsf{FEV}(\tau) \cup \{\hat{\alpha}\}) \subseteq \mathsf{dom}(\Gamma_R)$
then $\Delta = (\Delta_L * \Delta_R)$ *and* $(\Gamma_L * \Gamma_R) \xrightarrow{-*} (\Delta_L * \Delta_R)$.

(vi) *If* $\Gamma_L * \Gamma_R \vdash P \equiv Q \dashv \Delta$
and $\mathsf{FEV}(P) \cup \mathsf{FEV}(Q) \subseteq \mathsf{dom}(\Gamma_R)$
then $\Delta = (\Delta_L * \Delta_R)$ *and* $(\Gamma_L * \Gamma_R) \xrightarrow{-*} (\Delta_L * \Delta_R)$.

(vii) *If* $\Gamma_L * \Gamma_R \vdash A \equiv B \dashv \Delta$
and $\mathsf{FEV}(A) \cup \mathsf{FEV}(B) \subseteq \mathsf{dom}(\Gamma_R)$
then $\Delta = (\Delta_L * \Delta_R)$ *and* $(\Gamma_L * \Gamma_R) \xrightarrow{-*} (\Delta_L * \Delta_R)$.

Lemma 71 (Separation for Subtyping). *Go to proof*
If $\Gamma_L * \Gamma_R \vdash A <:^{\pm} B \dashv \Delta$
and $\mathsf{FEV}(A) \subseteq \mathsf{dom}(\Gamma_R)$
and $\mathsf{FEV}(B) \subseteq \mathsf{dom}(\Gamma_R)$
then $\Delta = (\Delta_L * \Delta_R)$ *and* $(\Gamma_L * \Gamma_R) \xrightarrow{-*} (\Delta_L * \Delta_R)$.

Lemma 72 (Separation—Main). *Go to proof*

(Spines) *If* $\Gamma_L * \Gamma_R \vdash s : A\ p \gg C\ q \dashv \Delta$
or $\Gamma_L * \Gamma_R \vdash s : A\ p \gg C\ \lceil q \rceil \dashv \Delta$
and $\Gamma_L * \Gamma_R \vdash A\ p\ \mathit{type}$
and $\mathsf{FEV}(A) \subseteq \mathsf{dom}(\Gamma_R)$
then $\Delta = (\Delta_L * \Delta_R)$ *and* $(\Gamma_L * \Gamma_R) \xrightarrow{-*} (\Delta_L * \Delta_R)$ *and* $\mathsf{FEV}(C) \subseteq \mathsf{dom}(\Delta_R)$.

(Checking) *If* $\Gamma_L * \Gamma_R \vdash e \Leftarrow C\ p \dashv \Delta$
and $\Gamma_L * \Gamma_R \vdash C\ p\ \mathit{type}$
and $\mathsf{FEV}(C) \subseteq \mathsf{dom}(\Gamma_R)$
then $\Delta = (\Delta_L * \Delta_R)$ *and* $(\Gamma_L * \Gamma_R) \xrightarrow{-*} (\Delta_L * \Delta_R)$.

(Synthesis) *If* $\Gamma_L * \Gamma_R \vdash e \Rightarrow A\ p \dashv \Delta$
then $\Delta = (\Delta_L * \Delta_R)$ *and* $(\Gamma_L * \Gamma_R) \xrightarrow{-*} (\Delta_L * \Delta_R)$.

(Match) *If* $\Gamma_L * \Gamma_R \vdash \Pi :: \vec{A} \Leftarrow C\ p \dashv \Delta$
and $\mathsf{FEV}(\vec{A}) = \emptyset$
and $\mathsf{FEV}(C) \subseteq \mathsf{dom}(\Gamma_R)$
then $\Delta = (\Delta_L * \Delta_R)$ *and* $(\Gamma_L * \Gamma_R) \xrightarrow{-*} (\Delta_L * \Delta_R)$.

(Match Elim.) *If* $\Gamma_L * \Gamma_R \;/\; P \vdash \Pi :: \vec{A} \Leftarrow C\ p \dashv \Delta$
and $\mathsf{FEV}(P) = \emptyset$
and $\mathsf{FEV}(\vec{A}) = \emptyset$
and $\mathsf{FEV}(C) \subseteq \mathsf{dom}(\Gamma_R)$
then $\Delta = (\Delta_L * \Delta_R)$ *and* $(\Gamma_L * \Gamma_R) \xrightarrow{*} (\Delta_L * \Delta_R)$.

H Decidability of Algorithmic Subtyping

Definition 6. *The following connectives are* large*:*

$$\forall \quad \supset \quad \wedge$$

A type is large *iff its head connective is large. (Note that a non-large type may contain large connectives, provided they are not in head position.)*

The number of these connectives in a type A *is denoted by* $\#\mathsf{large}(A)$.

H.1 Lemmas for Decidability of Subtyping

Lemma 73 (Substitution Isn't Large)**.** *Go to proof*
For all contexts Θ*, we have* $\#\mathsf{large}([\Theta]A) = \#\mathsf{large}(A)$.

Lemma 74 (Instantiation Solves)**.** *Go to proof*
If $\Gamma \vdash \hat{\alpha} := \tau : \kappa \dashv \Delta$ *and* $[\Gamma]\tau = \tau$ *and* $\hat{\alpha} \notin \mathrm{FV}([\Gamma]\tau)$ *then* $|\mathsf{unsolved}(\Gamma)| = |\mathsf{unsolved}(\Delta)| + 1$.

Lemma 75 (Checkeq Solving)**.** *Go to proof* *If* $\Gamma \vdash s \doteq t : \kappa \dashv \Delta$ *then either* $\Delta = \Gamma$ *or* $|\mathsf{unsolved}(\Delta)| < |\mathsf{unsolved}(\Gamma)|$.

Lemma 76 (Prop Equiv Solving)**.** *Go to proof*
If $\Gamma \vdash P \equiv Q \dashv \Delta$ *then either* $\Delta = \Gamma$ *or* $|\mathsf{unsolved}(\Delta)| < |\mathsf{unsolved}(\Gamma)|$.

Lemma 77 (Equiv Solving)**.** *Go to proof*
If $\Gamma \vdash A \equiv B \dashv \Delta$ *then either* $\Delta = \Gamma$ *or* $|\mathsf{unsolved}(\Delta)| < |\mathsf{unsolved}(\Gamma)|$.

Lemma 78 (Decidability of Propositional Judgments)**.** *Go to proof*
The following judgments are decidable, with Δ *as output in (1)–(3), and* $\Delta^\perp$ *as output in (4) and (5).*

We assume $\sigma = [\Gamma]\sigma$ *and* $t = [\Gamma]t$ *in (1) and (4). Similarly, in the other parts we assume* $P = [\Gamma]P$ *and (in part (3))* $Q = [\Gamma]Q$.

(1) $\Gamma \vdash \sigma \doteq t : \kappa \dashv \Delta$

(2) $\Gamma \vdash P\ \mathit{true} \dashv \Delta$

(3) $\Gamma \vdash P \equiv Q \dashv \Delta$

(4) $\Gamma \;/\; \sigma \doteq t : \kappa \dashv \Delta^\perp$

(5) $\Gamma \;/\; P \dashv \Delta^\perp$

Lemma 79 (Decidability of Equivalence)**.** *Go to proof*
Given a context Γ *and types* A, B *such that* $\Gamma \vdash A$ *type and* $\Gamma \vdash B$ *type and* $[\Gamma]A = A$ *and* $[\Gamma]B = B$*, it is decidable whether there exists* Δ *such that* $\Gamma \vdash A \equiv B \dashv \Delta$.

H.2 Decidability of Subtyping

Theorem 1 (Decidability of Subtyping)**.** *Go to proof*
Given a context Γ *and types* A, B *such that* $\Gamma \vdash A$ *type and* $\Gamma \vdash B$ *type and* $[\Gamma]A = A$ *and* $[\Gamma]B = B$*, it is decidable whether there exists* Δ *such that* $\Gamma \vdash A <:^{\pm} B \dashv \Delta$.

H.3 Decidability of Matching and Coverage

Lemma 80 (Decidability of Expansion Judgments). *Go to proof*
Given branches Π, it is decidable whether:

(1) there exists Π' such that $\Pi \overset{\times}{\rightsquigarrow} \Pi'$;

(2) there exist Π_L and Π_R such that $\Pi \overset{+}{\rightsquigarrow} \Pi_L \parallel \Pi_R$;

(3) there exists Π' such that $\Pi \overset{\text{var}}{\rightsquigarrow} \Pi'$;

(4) there exists Π' such that $\Pi \overset{1}{\rightsquigarrow} \Pi'$.

Theorem 2 (Decidability of Coverage). *Go to proof*
Given a context Γ, branches Π and types $\vec{A}$, it is decidable whether $\Gamma \vdash \Pi$ covers $\vec{A}$ is derivable.

H.4 Decidability of Typing

Theorem 3 (Decidability of Typing). *Go to proof*

(i) Synthesis: *Given a context Γ, a principality p, and a term e,*
it is decidable whether there exist a type A and a context Δ such that
$\Gamma \vdash e \Rightarrow A\ p \dashv \Delta$.

(ii) Spines: *Given a context Γ, a spine s, a principality p, and a type A such that $\Gamma \vdash A$ type,*
it is decidable whether there exist a type B, a principality q and a context Δ such that
$\Gamma \vdash s : A\ p \gg B\ q \dashv \Delta$.

(iii) Checking: *Given a context Γ, a principality p, a term e, and a type B such that $\Gamma \vdash B$ type,*
it is decidable whether there is a context Δ such that
$\Gamma \vdash e \Leftarrow B\ p \dashv \Delta$.

(iv) Matching: *Given a context Γ, branches Π, a list of types $\vec{A}$, a type C, and a principality p, it is decidable whether there exists Δ such that $\Gamma \vdash \Pi :: \vec{A} \Leftarrow C\ p \dashv \Delta$.*

Also, if given a proposition P as well, it is decidable whether there exists Δ such that $\Gamma \,/\, P \vdash \Pi :: \vec{A} \Leftarrow C\ p \dashv \Delta$.

I Determinacy

Lemma 81 (Determinacy of Auxiliary Judgments). *Go to proof*

(1) Elimeq: *Given Γ, σ, t, κ such that $\mathsf{FEV}(\sigma) \cup \mathsf{FEV}(t) = \emptyset$ and $\mathcal{D}_1 :: \Gamma \,/\, \sigma \triangleq t : \kappa \dashv \Delta_1^\perp$ and $\mathcal{D}_2 :: \Gamma \,/\, \sigma \triangleq t : \kappa \dashv \Delta_2^\perp$,*
it is the case that $\Delta_1^\perp = \Delta_2^\perp$.

(2) Instantiation: *Given Γ, $\hat{\alpha}$, t, κ such that $\hat{\alpha} \in \mathsf{unsolved}(\Gamma)$ and $\Gamma \vdash t : \kappa$ and $\hat{\alpha} \notin \mathsf{FV}(t)$*
and $\mathcal{D}_1 :: \Gamma \vdash \hat{\alpha} := t : \kappa \dashv \Delta_1$ and $\mathcal{D}_2 :: \Gamma \vdash \hat{\alpha} := t : \kappa \dashv \Delta_2$
it is the case that $\Delta_1 = \Delta_2$.

(3) Symmetric instantiation:

Given Γ, $\hat{\alpha}$, $\hat{\beta}$, κ such that $\hat{\alpha}, \hat{\beta} \in \mathsf{unsolved}(\Gamma)$ and $\hat{\alpha} \neq \hat{\beta}$
and $\mathcal{D}_1 :: \Gamma \vdash \hat{\alpha} := \hat{\beta} : \kappa \dashv \Delta_1$ and $\mathcal{D}_2 :: \Gamma \vdash \hat{\beta} := \hat{\alpha} : \kappa \dashv \Delta_2$
it is the case that $\Delta_1 = \Delta_2$.

(4) Checkeq: *Given Γ, σ, t, κ such that $\mathcal{D}_1 :: \Gamma \vdash \sigma \triangleq t : \kappa \dashv \Delta_1$ and $\mathcal{D}_2 :: \Gamma \vdash \sigma \triangleq t : \kappa \dashv \Delta_2$*
it is the case that $\Delta_1 = \Delta_2$.

(5) Elimprop: *Given Γ, P such that $\mathcal{D}_1 :: \Gamma \,/\, P \dashv \Delta_1^\perp$ and $\mathcal{D}_2 :: \Gamma \,/\, P \dashv \Delta_2^\perp$*
it is the case that $\Delta_1 = \Delta_2$.

(6) Checkprop: *Given Γ, P such that $\mathcal{D}_1 :: \Gamma \vdash P$ true $\dashv \Delta_1$ and $\mathcal{D}_2 :: \Gamma \vdash P$ true $\dashv \Delta_2$,*
it is the case that $\Delta_1 = \Delta_2$.

Lemma 82 (Determinacy of Equivalence). *Go to proof*

(1) Propositional equivalence: *Given* Γ, P, Q *such that* $\mathcal{D}_1 :: \Gamma \vdash P \equiv Q \dashv \Delta_1$ *and* $\mathcal{D}_2 :: \Gamma \vdash P \equiv Q \dashv \Delta_2$, *it is the case that* $\Delta_1 = \Delta_2$.

(2) Type equivalence: *Given* Γ, A, B *such that* $\mathcal{D}_1 :: \Gamma \vdash A \equiv B \dashv \Delta_1$ *and* $\mathcal{D}_2 :: \Gamma \vdash A \equiv B \dashv \Delta_2$, *it is the case that* $\Delta_1 = \Delta_2$.

Theorem 4 (Determinacy of Subtyping). *Go to proof*

(1) Subtyping: *Given* Γ, e, A, B *such that* $\mathcal{D}_1 :: \Gamma \vdash A <:^{\pm} B \dashv \Delta_1$ *and* $\mathcal{D}_2 :: \Gamma \vdash A <:^{\pm} B \dashv \Delta_2$, *it is the case that* $\Delta_1 = \Delta_2$.

Theorem 5 (Determinacy of Typing). *Go to proof*

(1) Checking: *Given* Γ, e, A, p *such that* $\mathcal{D}_1 :: \Gamma \vdash e \Leftarrow A\ p \dashv \Delta_1$ *and* $\mathcal{D}_2 :: \Gamma \vdash e \Leftarrow A\ p \dashv \Delta_2$, *it is the case that* $\Delta_1 = \Delta_2$.

(2) Synthesis: *Given* Γ, e *such that* $\mathcal{D}_1 :: \Gamma \vdash e \Rightarrow B_1\ p_1 \dashv \Delta_1$ *and* $\mathcal{D}_2 :: \Gamma \vdash e \Rightarrow B_2\ p_2 \dashv \Delta_2$, *it is the case that* $B_1 = B_2$ *and* $p_1 = p_2$ *and* $\Delta_1 = \Delta_2$.

(3) Spine judgments:

Given Γ, e, A, p *such that* $\mathcal{D}_1 :: \Gamma \vdash e : A\ p \gg C_1\ q_1 \dashv \Delta_1$ *and* $\mathcal{D}_2 :: \Gamma \vdash e : A\ p \gg C_2\ q_2 \dashv \Delta_2$, *it is the case that* $C_1 = C_2$ *and* $q_1 = q_2$ *and* $\Delta_1 = \Delta_2$.

The same applies for derivations of the principality-recovering judgments $\Gamma \vdash e : A\ p \gg C_k\ \lceil q_k \rceil \dashv \Delta_k$.

(4) Match judgments:

Given Γ, Π, $\vec{A}$, p, C *such that* $\mathcal{D}_1 :: \Gamma \vdash \Pi :: \vec{A} \Leftarrow C\ p \dashv \Delta_1$ *and* $\mathcal{D}_2 :: \Gamma \vdash \Pi :: \vec{A} \Leftarrow C\ p \dashv \Delta_2$, *it is the case that* $\Delta_1 = \Delta_2$.

Given Γ, P, Π, $\vec{A}$, p, C *such that* $\mathcal{D}_1 :: \Gamma\ /\ P \vdash \Pi :: \vec{A} \Leftarrow C\ p \dashv \Delta_1$ *and* $\mathcal{D}_2 :: \Gamma\ /\ P \vdash \Pi :: \vec{A} \Leftarrow C\ p \dashv \Delta_2$, *it is the case that* $\Delta_1 = \Delta_2$.

J Soundness

J.1 Soundness of Instantiation

Lemma 83 (Soundness of Instantiation). *Go to proof*
If $\Gamma \vdash \hat{\alpha} := \tau : \kappa \dashv \Delta$ *and* $\hat{\alpha} \notin FV([\Gamma]\tau)$ *and* $[\Gamma]\tau = \tau$ *and* $\Delta \longrightarrow \Omega$ *then* $[\Omega]\hat{\alpha} = [\Omega]\tau$.

J.2 Soundness of Checkeq

Lemma 84 (Soundness of Checkeq). *Go to proof*
If $\Gamma \vdash \sigma \doteq t : \kappa \dashv \Delta$ *where* $\Delta \longrightarrow \Omega$ *then* $[\Omega]\sigma = [\Omega]t$.

J.3 Soundness of Equivalence (Propositions and Types)

Lemma 85 (Soundness of Propositional Equivalence). *Go to proof*
If $\Gamma \vdash P \equiv Q \dashv \Delta$ *where* $\Delta \longrightarrow \Omega$ *then* $[\Omega]P = [\Omega]Q$.

Lemma 86 (Soundness of Algorithmic Equivalence). *Go to proof*
If $\Gamma \vdash A \equiv B \dashv \Delta$ *where* $\Delta \longrightarrow \Omega$ *then* $[\Omega]A = [\Omega]B$.

J.4 Soundness of Checkprop

Lemma 87 (Soundness of Checkprop). *Go to proof*
If $\Gamma \vdash P$ *true* $\dashv \Delta$ *and* $\Delta \longrightarrow \Omega$ *then* $\Psi \vdash [\Omega]P$ *true*.

J.5 Soundness of Eliminations (Equality and Proposition)

Lemma 88 (Soundness of Equality Elimination). *Go to proof*
If $[\Gamma]\sigma = \sigma$ *and* $[\Gamma]t = t$ *and* $\Gamma \vdash \sigma : \kappa$ *and* $\Gamma \vdash t : \kappa$ *and* $\mathsf{FEV}(\sigma) \cup \mathsf{FEV}(t) = \emptyset$*, then:*

(1) If $\Gamma \;/\; \sigma \doteq t : \kappa \dashv \Delta$
then $\Delta = (\Gamma, \Theta)$ *where* $\Theta = (\alpha_1 = t_1, \ldots, \alpha_n = t_n)$ *and*
for all Ω *such that* $\Gamma \longrightarrow \Omega$
and all t' *such that* $\Omega \vdash t' : \kappa'$*,*
it is the case that $[\Omega, \Theta]t' = [\theta][\Omega]t'$*, where* $\theta = \mathsf{mgu}(\sigma, t)$*.*

(2) If $\Gamma \;/\; \sigma \doteq t : \kappa \dashv \bot$ *then* $\mathsf{mgu}(\sigma, t) = \bot$ *(that is, no most general unifier exists).*

J.6 Soundness of Subtyping

Theorem 6 (Soundness of Algorithmic Subtyping). *Go to proof*
If $[\Gamma]A = A$ *and* $[\Gamma]B = B$ *and* $\Gamma \vdash A$ *type and* $\Gamma \vdash B$ *type and* $\Delta \longrightarrow \Omega$ *and* $\Gamma \vdash A <:^{\pm} B \dashv \Delta$ *then* $[\Omega]\Delta \vdash [\Omega]A \leq^{\pm} [\Omega]B$*.*

J.7 Soundness of Typing

Theorem 7 (Soundness of Match Coverage). *Go to proof*

1. *If* $\Gamma \vdash \Pi$ *covers* $\vec{A}$ *and* $\Gamma \longrightarrow \Omega$ *and* $\Gamma \vdash \vec{A}$ **!** *types and* $[\Gamma]\vec{A} = \vec{A}$ *then* $[\Omega]\Gamma \vdash \Pi$ *covers* $\vec{A}$*.*
2. *If* $\Gamma \;/\; P \vdash \Pi$ *covers* $\vec{A}$ *and* $\Gamma \longrightarrow \Omega$ *and* $\Gamma \vdash \vec{A}$ **!** *types and* $[\Gamma]\vec{A} = \vec{A}$ *and* $[\Gamma]P = P$ *then* $[\Omega]\Gamma \;/\; P \vdash \Pi$ *covers* $\vec{A}$*.*

Lemma 89 (Well-formedness of Algorithmic Typing). *Go to proof*
Given Γ *ctx:*

(i) If $\Gamma \vdash e \Rightarrow A\ p \dashv \Delta$ *then* $\Delta \vdash A\ p$ *type.*

(ii) If $\Gamma \vdash s : A\ p \gg B\ q \dashv \Delta$ *and* $\Gamma \vdash A\ p$ *type then* $\Delta \vdash B\ q$ *type.*

Definition 7 (Measure). *Let measure* $\mathcal{M}$ *on typing judgments be a lexicographic ordering:*

1. *first, the subject expression* e*, spine* s*, or matches* Π*—regarding all types in annotations as equal in size;*
2. *second, the partial order on judgment forms where an ordinary spine judgment is* smaller *than a principality-recovering spine judgment—and with all other judgment forms considered equal in size; and,*
3. *third, the derivation height.*

$$\left\langle e/s/\Pi, \begin{array}{c}\text{ordinary spine judgment}\\ < \\ \text{recovering spine judgment}\end{array}, \ \mathrm{height}(\mathcal{D}) \right\rangle$$

Note that this definition doesn't take notice of whether a spine judgment is declarative or algorithmic.

This measure works to show soundness and completeness. We list each rule below, along with a 3-tuple. For example, for Sub we write $\langle =, =, < \rangle$, meaning that each judgment to which we need to apply the i.h. has a subject of the same size ($=$), a judgment form of the same size ($=$), and a smaller derivation height. We write $-$ when a part of the measure need not be considered because a lexicographically more significant part is smaller, as in the Anno rule, where the premise has a smaller subject: $\langle <, -, - \rangle$.

Algorithmic rules (soundness cases):

- Var, 1I, 1I$\hat{\alpha}$, EmptySpine and Nil have no premises, or only auxiliary judgments as premises.
- Sub: $\langle =, =, < \rangle$

- Anno: $\langle <, -, - \rangle$
- $\forall$I, $\forall$Spine, $\wedge$I: $\langle =, =, < \rangle$
- $\supset$I: $\langle =, =, < \rangle$
- $\supset$I$\bot$ has only an auxiliary judgment, to which we need not apply the i.h., putting it in the same class as the rules with no premises.
- $\supset$Spine: $\langle =, =, < \rangle$
- $\rightarrow$I, $\rightarrow$I$\hat{\alpha}$, $\rightarrow$E, Rec: $\langle <, -, - \rangle$
- SpineRecover: $\langle =, <, - \rangle$
- SpinePass: $\langle =, <, - \rangle$
- $\rightarrow$Spine, $+\mathsf{I}_k$, $+\mathsf{I}\hat{\alpha}_k$, $\times$I, $\times$I$\hat{\alpha}$, Cons: $\langle <, -, - \rangle$
- $\hat{\alpha}$Spine: $\langle =, =, < \rangle$
- Case: $\langle <, -, - \rangle$

Declarative rules (completeness cases):

- DeclVar, Decl1I, DeclEmptySpine and DeclNil have no premises, or only auxiliary judgments as premises.
- DeclSub: $\langle =, =, < \rangle$
- DeclAnno: $\langle <, -, - \rangle$
- Decl$\forall$I, Decl$\forall$Spine, Decl$\wedge$I, Decl$\supset$I, Decl$\supset$Spine: $\langle =, =, < \rangle$
- Decl$\rightarrow$I, Decl$\rightarrow$E, DeclRec: $\langle <, -, - \rangle$
- DeclSpineRecover: $\langle =, <, - \rangle$
- DeclSpinePass: $\langle =, <, - \rangle$
- Decl$\rightarrow$Spine, Decl$+\mathsf{I}_k$, Decl$\times$I, DeclCase, DeclCons, $\langle <, -, - \rangle$

Theorem 8 (Soundness of Algorithmic Typing)**.** *Go to proof*
Given $\Delta \longrightarrow \Omega$*:*

(i) *If* $\Gamma \vdash e \Leftarrow A\ p \dashv \Delta$ *and* $\Gamma \vdash A\ p$ *type then* $[\Omega]\Delta \vdash [\Omega]e \Leftarrow [\Omega]A\ p$.

(ii) *If* $\Gamma \vdash e \Rightarrow A\ p \dashv \Delta$ *then* $[\Omega]\Delta \vdash [\Omega]e \Rightarrow [\Omega]A\ p$.

(iii) *If* $\Gamma \vdash s : A\ p \gg B\ q \dashv \Delta$ *and* $\Gamma \vdash A\ p$ *type then* $[\Omega]\Delta \vdash [\Omega]s : [\Omega]A\ p \gg [\Omega]B\ q$.

(iv) *If* $\Gamma \vdash s : A\ p \gg B\ \lceil q \rceil \dashv \Delta$ *and* $\Gamma \vdash A\ p$ *type then* $[\Omega]\Delta \vdash [\Omega]s : [\Omega]A\ p \gg [\Omega]B\ \lceil q \rceil$.

(v) *If* $\Gamma \vdash \Pi :: \vec{A} \Leftarrow C\ p \dashv \Delta$ *and* $\Gamma \vdash \vec{A}$ **!** *types and* $[\Gamma]\vec{A} = \vec{A}$ *and* $\Gamma \vdash C\ p$ *type*
then $[\Omega]\Delta \vdash [\Omega]\Pi :: [\Omega]\vec{A} \Leftarrow [\Omega]C\ p$.

(vi) *If* $\Gamma \ / \ P \vdash \Pi :: \vec{A} \Leftarrow C\ p \dashv \Delta$ *and* $\Gamma \vdash P$ *prop and* $\mathsf{FEV}(P) = \emptyset$ *and* $[\Gamma]P = P$
and $\Gamma \vdash \vec{A}$ **!** *types and* $\Gamma \vdash C\ p$ *type*
then $[\Omega]\Delta \ / \ [\Omega]P \vdash [\Omega]\Pi :: [\Omega]\vec{A} \Leftarrow [\Omega]C\ p$.

K Completeness

K.1 Completeness of Auxiliary Judgments

Lemma 90 (Completeness of Instantiation). *Go to proof*
Given $\Gamma \longrightarrow \Omega$ *and* $\mathsf{dom}(\Gamma) = \mathsf{dom}(\Omega)$ *and* $\Gamma \vdash \tau : \kappa$ *and* $\tau = [\Gamma]\tau$ *and* $\hat{\alpha} \in \mathsf{unsolved}(\Gamma)$ *and* $\hat{\alpha} \notin \mathrm{FV}(\tau)$*:*
If $[\Omega]\hat{\alpha} = [\Omega]\tau$
then there are Δ, Ω' *such that* $\Omega \longrightarrow \Omega'$ *and* $\Delta \longrightarrow \Omega'$ *and* $\mathsf{dom}(\Delta) = \mathsf{dom}(\Omega')$ *and* $\Gamma \vdash \hat{\alpha} := \tau : \kappa \dashv \Delta$.

Lemma 91 (Completeness of Checkeq). *Go to proof*
Given $\Gamma \longrightarrow \Omega$ *and* $\mathsf{dom}(\Gamma) = \mathsf{dom}(\Omega)$
and $\Gamma \vdash \sigma : \kappa$ *and* $\Gamma \vdash \tau : \kappa$
and $[\Omega]\sigma = [\Omega]\tau$
then $\Gamma \vdash [\Gamma]\sigma \triangleq [\Gamma]\tau : \kappa \dashv \Delta$
where $\Delta \longrightarrow \Omega'$ *and* $\mathsf{dom}(\Delta) = \mathsf{dom}(\Omega')$ *and* $\Omega \longrightarrow \Omega'$.

Lemma 92 (Completeness of Elimeq). *Go to proof*
If $[\Gamma]\sigma = \sigma$ *and* $[\Gamma]t = t$ *and* $\Gamma \vdash \sigma : \kappa$ *and* $\Gamma \vdash t : \kappa$ *and* $\mathsf{FEV}(\sigma) \cup \mathsf{FEV}(t) = \emptyset$ *then:*

(1) *If* $\mathsf{mgu}(\sigma, t) = \theta$
then $\Gamma \,/\, \sigma \triangleq t : \kappa \dashv (\Gamma, \Delta)$
where Δ *has the form* $\alpha_1 = t_1, \ldots, \alpha_n = t_n$
and for all u *such that* $\Gamma \vdash u : \kappa$*, it is the case that* $[\Gamma, \Delta]u = \theta([\Gamma]u)$.

(2) *If* $\mathsf{mgu}(\sigma, t) = \bot$ *(that is, no most general unifier exists) then* $\Gamma \,/\, \sigma \triangleq t : \kappa \dashv \bot$.

Lemma 93 (Substitution Upgrade). *Go to proof*
If Δ *has the form* $\alpha_1 = t_1, \ldots, \alpha_n = t_n$
and, for all u *such that* $\Gamma \vdash u : \kappa$*, it is the case that* $[\Gamma, \Delta]u = \theta([\Gamma]u)$,
then:

(i) *If* $\Gamma \vdash A$ *type then* $[\Gamma, \Delta]A = \theta([\Gamma]A)$.

(ii) *If* $\Gamma \longrightarrow \Omega$ *then* $[\Omega]\Gamma = \theta([\Omega]\Gamma)$.

(iii) *If* $\Gamma \longrightarrow \Omega$ *then* $[\Omega, \Delta](\Gamma, \Delta) = \theta([\Omega]\Gamma)$.

(iv) *If* $\Gamma \longrightarrow \Omega$ *then* $[\Omega, \Delta]e = \theta([\Omega]e)$.

Lemma 94 (Completeness of Propequiv). *Go to proof*
Given $\Gamma \longrightarrow \Omega$
and $\Gamma \vdash P$ *prop and* $\Gamma \vdash Q$ *prop*
and $[\Omega]P = [\Omega]Q$
then $\Gamma \vdash [\Gamma]P \equiv [\Gamma]Q \dashv \Delta$
where $\Delta \longrightarrow \Omega'$ *and* $\Omega \longrightarrow \Omega'$.

Lemma 95 (Completeness of Checkprop). *Go to proof*
If $\Gamma \longrightarrow \Omega$ *and* $\mathsf{dom}(\Gamma) = \mathsf{dom}(\Omega)$
and $\Gamma \vdash P$ *prop*
and $[\Gamma]P = P$
and $[\Omega]\Gamma \vdash [\Omega]P$ *true*
then $\Gamma \vdash P$ *true* $\dashv \Delta$
where $\Delta \longrightarrow \Omega'$ *and* $\Omega \longrightarrow \Omega'$ *and* $\mathsf{dom}(\Delta) = \mathsf{dom}(\Omega')$.

K.2 Completeness of Equivalence and Subtyping

Lemma 96 (Completeness of Equiv). *Go to proof*
If $\Gamma \longrightarrow \Omega$ *and* $\Gamma \vdash A$ *type and* $\Gamma \vdash B$ *type*
and $[\Omega]A = [\Omega]B$
then there exist Δ *and* Ω' *such that* $\Delta \longrightarrow \Omega'$ *and* $\Omega \longrightarrow \Omega'$ *and* $\Gamma \vdash [\Gamma]A \equiv [\Gamma]B \dashv \Delta$.

Theorem 9 (Completeness of Subtyping). *Go to proof*
If $\Gamma \longrightarrow \Omega$ *and* $\mathsf{dom}(\Gamma) = \mathsf{dom}(\Omega)$ *and* $\Gamma \vdash A$ *type and* $\Gamma \vdash B$ *type*
and $[\Omega]\Gamma \vdash [\Omega]A \leq^{\pm} [\Omega]B$
then there exist Δ *and* Ω' *such that* $\Delta \longrightarrow \Omega'$
and $\mathsf{dom}(\Delta) = \mathsf{dom}(\Omega')$
and $\Omega \longrightarrow \Omega'$
and $\Gamma \vdash [\Gamma]A <:^{\pm} [\Gamma]B \dashv \Delta$.

K.3 Completeness of Typing

Theorem 10 (Completeness of Match Coverage). *Go to proof*

1. *If* $[\Omega]\Gamma \vdash [\Omega]\Pi$ *covers* $[\Omega]\vec{A}$ *and* $\Gamma \longrightarrow \Omega$ *and* $\Gamma \vdash \vec{A}$ **!** *types and* $[\Gamma]\vec{A} = \vec{A}$
   then $\Gamma \vdash \Pi$ *covers* $\vec{A}$.
2. *If* $[\Omega]\Gamma \,/\, [\Omega]P \vdash [\Omega]\Pi$ *covers* $[\Omega]\vec{A}$ *and* $\Gamma \longrightarrow \Omega$ *and* $\Gamma \vdash \vec{A}$ **!** *types and* $[\Gamma]\vec{A} = \vec{A}$ *and* $[\Gamma]P = P$
   then $\Gamma \,/\, P \vdash \Pi$ *covers* $\vec{A}$.

Theorem 11 (Completeness of Algorithmic Typing). *Go to proof* *Given* $\Gamma \longrightarrow \Omega$ *such that* $\mathsf{dom}(\Gamma) = \mathsf{dom}(\Omega)$*:*

(i) *If* $\Gamma \vdash A\ p$ *type and* $[\Omega]\Gamma \vdash [\Omega]e \Leftarrow [\Omega]A\ p$ *and* $p' \sqsubseteq p$
then there exist Δ *and* Ω'
such that $\Delta \longrightarrow \Omega'$ *and* $\mathsf{dom}(\Delta) = \mathsf{dom}(\Omega')$ *and* $\Omega \longrightarrow \Omega'$
and $\Gamma \vdash e \Leftarrow [\Gamma]A\ p' \dashv \Delta$.

(ii) *If* $\Gamma \vdash A\ p$ *type and* $[\Omega]\Gamma \vdash [\Omega]e \Rightarrow A\ p$
then there exist Δ, Ω', A', *and* $p' \sqsubseteq p$
such that $\Delta \longrightarrow \Omega'$ *and* $\mathsf{dom}(\Delta) = \mathsf{dom}(\Omega')$ *and* $\Omega \longrightarrow \Omega'$
and $\Gamma \vdash e \Rightarrow A'\ p' \dashv \Delta$ *and* $A' = [\Delta]A'$ *and* $A = [\Omega']A'$.

(iii) *If* $\Gamma \vdash A\ p$ *type and* $[\Omega]\Gamma \vdash [\Omega]s : [\Omega]A\ p \gg B\ q$ *and* $p' \sqsubseteq p$
then there exist Δ, Ω', B' *and* $q' \sqsubseteq q$
such that $\Delta \longrightarrow \Omega'$ *and* $\mathsf{dom}(\Delta) = \mathsf{dom}(\Omega')$ *and* $\Omega \longrightarrow \Omega'$
and $\Gamma \vdash s : [\Gamma]A\ p' \gg B'\ q' \dashv \Delta$ *and* $B' = [\Delta]B'$ *and* $B = [\Omega']B'$.

(iv) *If* $\Gamma \vdash A\ p$ *type and* $[\Omega]\Gamma \vdash [\Omega]s : [\Omega]A\ p \gg B\ \lceil q \rceil$ *and* $p' \sqsubseteq p$
then there exist Δ, Ω', B', *and* $q' \sqsubseteq q$
such that $\Delta \longrightarrow \Omega'$ *and* $\mathsf{dom}(\Delta) = \mathsf{dom}(\Omega')$ *and* $\Omega \longrightarrow \Omega'$
and $\Gamma \vdash s : [\Gamma]A\ p' \gg B'\ \lceil q' \rceil \dashv \Delta$ *and* $B' = [\Delta]B'$ *and* $B = [\Omega']B'$.

(v) *If* $\Gamma \vdash \vec{A}$ **!** *types and* $\Gamma \vdash C\ p$ *type and* $[\Omega]\Gamma \vdash [\Omega]\Pi :: [\Omega]\vec{A} \Leftarrow [\Omega]C\ p$ *and* $p' \sqsubseteq p$
then there exist Δ, Ω', *and* C
such that $\Delta \longrightarrow \Omega'$ *and* $\mathsf{dom}(\Delta) = \mathsf{dom}(\Omega')$ *and* $\Omega \longrightarrow \Omega'$
and $\Gamma \vdash \Pi :: [\Gamma]\vec{A} \Leftarrow [\Gamma]C\ p' \dashv \Delta$.

(vi) *If* $\Gamma \vdash \vec{A}$ **!** *types and* $\Gamma \vdash P$ *prop and* $\mathsf{FEV}(P) = \emptyset$ *and* $\Gamma \vdash C\ p$ *type*
and $[\Omega]\Gamma \,/\, [\Omega]P \vdash [\Omega]\Pi :: [\Omega]\vec{A} \Leftarrow [\Omega]C\ p$
and $p' \sqsubseteq p$
then there exist Δ, Ω', *and* C
such that $\Delta \longrightarrow \Omega'$ *and* $\mathsf{dom}(\Delta) = \mathsf{dom}(\Omega')$ *and* $\Omega \longrightarrow \Omega'$
and $\Gamma \,/\, [\Gamma]P \vdash \Pi :: [\Gamma]\vec{A} \Leftarrow [\Gamma]C\ p' \dashv \Delta$.

Proofs

In the rest of this document, we prove the results stated above, with the same sectioning.

B′ Properties of the Declarative System

Lemma 1 (Declarative Well-foundedness)**.**
The inductive definition of the following judgments is well-founded:

(i) *synthesis* $\Psi \vdash e \Rightarrow B\ p$

(ii) *checking* $\Psi \vdash e \Leftarrow A\ p$

(iii) *checking, equality elimination* $\Psi\ /\ P \vdash e \Leftarrow C\ p$

(iv) *ordinary spine* $\Psi \vdash s : A\ p \gg B\ q$

(v) *recovery spine* $\Psi \vdash s : A\ p \gg B\ \lceil q \rceil$

(vi) *pattern matching* $\Psi \vdash \Pi :: \vec{A} \Leftarrow C\ p$

(vii) *pattern matching, equality elimination* $\Psi\ /\ P \vdash \Pi :: \vec{A} \Leftarrow C\ p$

Proof. Let $|e|$ be the size of the expression e. Let $|s|$ be the size of the spine s. Let $|\Pi|$ be the size of the branch list Π. Let $\#\mathsf{large}(A)$ be the number of "large" connectives $\forall$, $\exists$, $\supset$, $\wedge$ in A.

First, stratify judgments by the size of the term (expression, spine, or branches), and say that a judgment is *at* n if it types a term of size n. Order the main judgment forms as follows:

$$\begin{aligned} & \text{synthesis judgment at } n \\ < \ & \text{checking judgments at } n \\ < \ & \text{ordinary spine judgment at } n \\ < \ & \text{recovery spine judgment at } n \\ < \ & \text{match judgments at } n \\ < \ & \text{synthesis judgment at } n+1 \\ & \vdots \end{aligned}$$

Within the checking judgment forms at n, we compare types lexicographically, first by the number of large connectives, and then by the ordinary size. Within the match judgment forms at n, we compare using a lexicographic order of, first, $\#\mathsf{large}(\vec{A})$; second, the judgment form, considering the match judgment to be smaller than the matchelim judgment; third, the size of $\vec{A}$. These criteria order the judgments as follows:

$$\begin{aligned} & \text{synthesis judgment at } n \\ < \ & \big(\text{checking judgment at } n \text{ with } \#\mathsf{large}(A) = 1 \\ & < \ \text{checkelim judgment at } n \text{ with } \#\mathsf{large}(A) = 1 \\ & < \ \text{checking judgment at } n \text{ with } \#\mathsf{large}(A) = 2 \\ & < \ \text{checkelim judgment at } n \text{ with } \#\mathsf{large}(A) = 2 \\ & < \ \ldots\big) \\ < \ & \big(\text{match judgment at } n \text{ with } \#\mathsf{large}(\vec{A}) = 1 \text{ and } \vec{A} \text{ of size } 1 \\ & < \ \text{match judgment at } n \text{ with } \#\mathsf{large}(\vec{A}) = 1 \text{ and } \vec{A} \text{ of size } 2 \\ & < \ \text{matchelim judgment at } n \text{ with } \#\mathsf{large}(\vec{A}) = 1 \\ & < \ \text{match judgment at } n \text{ with } \#\mathsf{large}(\vec{A}) = 2 \text{ and } \vec{A} \text{ of size } 1 \\ & < \ \text{match judgment at } n \text{ with } \#\mathsf{large}(\vec{A}) = 2 \text{ and } \vec{A} \text{ of size } 2 \\ & < \ \text{matchelim judgment at } n \text{ with } \#\mathsf{large}(\vec{A}) = 2 \\ & < \ \ldots\big) \end{aligned}$$

The class of ordinary spine judgments at 1 need not be refined, because the only ordinary spine rule applicable to a spine of size 1 is DeclEmptySpine, which has no premises; rules Decl$\forall$Spine, Decl$\supset$Spine, and Decl$\rightarrow$Spine are restricted to non-empty spines and can only apply to larger terms.

Similarly, the class of match judgments at 1 need not be refined, because only DeclMatchEmpty is applicable.

Note that we distinguish the "checkelim" form $\Psi \;/\; P \vdash e \Leftarrow A\ p$ of the checking judgment. We also define the size of an expression e to consider all types in annotations to be of the same size, that is,

$$|(e : A)| \;=\; |e| + 1$$

Thus, $|\theta(e)| = |e|$, even when e has annotations. This is used for DeclCheckUnify; see below.

We assume that coverage, which does not depend on any other typing judgments, is well-founded. We likewise assume that subtyping, $\Psi \vdash A\ \mathit{type}$, $\Psi \vdash \tau : \kappa$, and $\Psi \vdash P\ \mathit{prop}$ are well-founded.

We now show that, for each class of judgments, every judgment in that class depends only on smaller judgments.

- **Synthesis judgments**

  Claim: For all n, synthesis at n depends only on judgments at $n-1$ or less.

  Proof. Rule DeclVar has no premises.
  Rule DeclAnno depends on a premise at a strictly smaller term.
  Rule Decl$\to$E depends on (1) a synthesis premise at a strictly smaller term, and (2) a recovery spine judgment at a strictly smaller term.

- **Checking judgments**

  Claim: For all $n \geq 1$, the checking judgment over terms of size n with type of size m depends only on

  (1) synthesis judgments at size n or smaller, and
  (2) checking judgments at size $n-1$ or smaller, and
  (3) checking judgments at size n with fewer large connectives, and
  (4) checkelim judgments at size n with fewer large connectives, and
  (5) match judgments at size $n-1$ or smaller.

  Proof. Rule DeclSub depends on a synthesis judgment of size n. (1)
  Rule Decl1I has no premises.
  Rule Decl$\forall$I depends on a checking judgment at n with fewer large connectives. (3)
  Rule Decl$\wedge$I depends on a checking judgment at n with fewer large connectives. (3)
  Rule Decl$\supset$I depends on a checkelim judgment at n with fewer large connectives. (4)
  Rules Decl$\to$I, DeclRec, Decl$+\mathsf{I}_k$, Decl$\times$I, and DeclCons depend on checking judgments at size $< n$. (2)
  Rule DeclNil depends only on an auxiliary judgment.
  Rule DeclCase depends on:

  - a synthesis judgment at size n (1),
  - a match judgment at size $< n$ (5), and
  - a coverage judgment.

- **Checkelim judgments**

  Claim: For all $n \geq 1$, the checkelim judgment $\Psi \;/\; P \vdash e \Leftarrow A\ p$ over terms of size n depends only on checking judgments at size n, with a type A' such that $\#\mathsf{large}(A') = \#\mathsf{large}(A)$.

  Proof. Rule DeclCheck$\perp$ has no nontrivial premises.
  Rule DeclCheckUnify depends on a checking judgment: Since $|\theta(e)| = |e|$, this checking judgment is at n. Since the mgu θ is over monotypes, $\#\mathsf{large}(\theta(A)) = \#\mathsf{large}(A)$.

- **Ordinary spine judgments**

  An ordinary spine judgment at 1 depends on no other judgments: the only spine of size 1 is the empty spine, so only DeclEmptySpine applies, and it has no premises.

  Claim: For all $n \geq 2$, the ordinary spine judgment $\Psi \vdash s : A\ p \gg C\ q$ over spines of size n depends only on

(a) checking judgments at size $n-1$ or smaller, and

(b) ordinary spine judgments at size $n-1$ or smaller, and

(c) ordinary spine judgments at size n with strictly smaller $\#\mathsf{large}(A)$.

Proof. Rule Decl$\forall$Spine depends on an ordinary spine judgment of size n, with a type that has fewer large connectives. (c)
Rule Decl$\supset$Spine depends on an ordinary spine judgment of size n, with a type that has fewer large connectives. (c)
Rule DeclEmptySpine has no premises.
Rule Decl$\rightarrow$Spine depends on a checking judgment of size $n-1$ or smaller (a) and an ordinary spine judgment of size $n-1$ or smaller (b).

- **Recovery spine judgments**

  Claim: For all n, the recovery spine judgment at n depends only on ordinary spine judgments at n.

  Proof. Rules DeclSpineRecover and DeclSpinePass depend only on ordinary spine judgments at n.

- **Match judgments**

  Claim: For all $n \geq 1$, the match judgment $\Psi \vdash \Pi :: \vec{A} \Leftarrow C\ p$ over Π of size n depends only on

  (a) checking judgments at size $n-1$ or smaller, and

  (b) match judgments at size $n-1$ or smaller, and

  (c) match judgments at size n with smaller $\vec{A}$, and

  (d) matchelim judgments at size n with fewer large connectives in $\vec{A}$.

  Proof. Rule DeclMatchEmpty has no premises.
  Rule DeclMatchSeq depends on match judgments at $n-1$ or smaller (b).
  Rule DeclMatchBase depends on a checking judgment at $n-1$ or smaller (a).
  Rules DeclMatch$\times$, DeclMatch$+_k$, DeclMatchNeg, and DeclMatchWild depend on match judgments at $n-1$ or smaller (b).
  Rule DeclMatch$\exists$ depends on a match judgment at size n with smaller $\vec{A}$ (c).
  Rule DeclMatch$\wedge$ depends on an matchelim judgment at n, with fewer large connectives in $\vec{A}$. (d)

- **Matchelim judgments**

  Claim: For all $n \geq 1$, the matchelim judgment $\Psi\ /\ \Pi \vdash P :: \vec{A} \Leftarrow C\ p$ over Ψ of size n depends only on match judgments with the same number of large connectives in $\vec{A}$.

  Proof. Rule DeclMatch$\perp$ has no nontrivial premises.
  Rule DeclMatchUnify depends on a match judgment with the same number of large connectives (similar to DeclCheckUnify, considered above). □

Lemma 2 (Declarative Weakening)**.**

(i) *If* $\Psi_0, \Psi_1 \vdash t : \kappa$ *then* $\Psi_0, \Psi, \Psi_1 \vdash t : \kappa$.

(ii) *If* $\Psi_0, \Psi_1 \vdash P\ prop$ *then* $\Psi_0, \Psi, \Psi_1 \vdash P\ prop$.

(iii) *If* $\Psi_0, \Psi_1 \vdash P\ true$ *then* $\Psi_0, \Psi, \Psi_1 \vdash P\ true$.

(iv) *If* $\Psi_0, \Psi_1 \vdash A\ type$ *then* $\Psi_0, \Psi, \Psi_1 \vdash A\ type$.

Proof. By induction on the derivation. □

Lemma 3 (Declarative Term Substitution)**.** *Suppose* $\Psi \vdash t : \kappa$. *Then:*

1. *If* $\Psi_0, \alpha : \kappa, \Psi_1 \vdash t' : \kappa$ *then* $\Psi_0, [t/\alpha]\Psi_1 \vdash [t/\alpha]t' : \kappa$.
2. *If* $\Psi_0, \alpha : \kappa, \Psi_1 \vdash P\ prop$ *then* $\Psi_0, [t/\alpha]\Psi_1 \vdash [t/\alpha]P\ prop$.
3. *If* $\Psi_0, \alpha : \kappa, \Psi_1 \vdash A\ type$ *then* $\Psi_0, [t/\alpha]\Psi_1 \vdash [t/\alpha]A\ type$.
4. *If* $\Psi_0, \alpha : \kappa, \Psi_1 \vdash A \leq^{\pm} B$ *then* $\Psi_0, [t/\alpha]\Psi_1 \vdash [t/\alpha]A \leq^{\pm} [t/\alpha]B$.

5. *If* $\Psi_0, \alpha : \kappa, \Psi_1 \vdash P$ *true then* $\Psi_0, [t/\alpha]\Psi_1 \vdash [t/\alpha]P$ *true.*

Proof. By induction on the derivation of the substitutee. □

Lemma 4 (Reflexivity of Declarative Subtyping)**.**
Given $\Psi \vdash A$ *type, we have that* $\Psi \vdash A \leq^{\pm} A$.

Proof. By induction on A, writing p for the sign of the subtyping judgment.

Our induction metric is the number of quantifiers on the outside of A, plus one if the polarity of A and the subtyping judgment do not match up (that is, if $neg(A)$ and $p = +$, or $pos(A)$ and $p = -$).

- **Case** $nonpos(A), nonneg(A), p = \pm$:

  By rule $\leq$Refl$\pm$.

- **Case** $A = \exists b : \kappa.\, B, p = +$:

  $\Psi, b : \kappa \vdash B \leq^{+} B$ By i.h. (one less quantifier)
  $\Psi, b : \kappa \vdash b : \kappa$ By rule UvarSort
  $\Psi, b : \kappa \vdash B \leq^{+} \exists b : \kappa.\, B$ By rule $\leq\exists$R
  $\Psi \vdash \exists b : \kappa.\, B \leq^{+} \exists b : \kappa.\, B$ By rule $\leq\exists$L

- **Case** $A = \exists b : \kappa.\, B, p = -$:

  $\Psi \vdash \exists b : \kappa.\, B \leq^{+} \exists b : \kappa.\, B$ By i.h. (polarities match)
  $\Psi \vdash \exists b : \kappa.\, B \leq^{-} \exists b : \kappa.\, B$ By $\leq^{+}_{-}$

- **Case** $A = \forall b : \kappa.\, B, p = +$:

  $\Psi \vdash \forall b : \kappa.\, B \leq^{-} \forall b : \kappa.\, B$ By i.h. (polarities match)
  $\Psi \vdash \forall b : \kappa.\, B \leq^{+} \forall b : \kappa.\, B$ By $\leq^{-}_{+}$

- **Case** $A = \forall b : \kappa.\, B, p = -$:

  $\Psi, b : \kappa \vdash B \leq^{-} B$ By i.h. (one less quantifier)
  $\Psi, b : \kappa \vdash b : \kappa$ By rule UvarSort
  $\Psi, b : \kappa \vdash \forall b : \kappa.\, B \leq^{-} B$ By rule $\leq\forall$L
  $\Psi \vdash \forall b : \kappa.\, B \leq^{-} \forall b : \kappa.\, B$ By rule $\leq\forall$R □

Lemma 5 (Subtyping Inversion)**.**

- *If* $\Psi \vdash \exists \alpha : \kappa.\, A \leq^{+} B$ *then* $\Psi, \alpha : \kappa \vdash A \leq^{+} B$.
- *If* $\Psi \vdash A \leq^{-} \forall \beta : \kappa.\, B$ *then* $\Psi, \beta : \kappa \vdash A \leq^{-} B$.

Proof. By a routine induction on the subtyping derivations. □

Lemma 6 (Subtyping Polarity Flip)**.**

- *If* $nonpos(A)$ *and* $nonpos(B)$ *and* $\Psi \vdash A \leq^{+} B$
  then $\Psi \vdash A \leq^{-} B$ *by a derivation of the same or smaller size.*
- *If* $nonneg(A)$ *and* $nonneg(B)$ *and* $\Psi \vdash A \leq^{-} B$
  then $\Psi \vdash A \leq^{+} B$ *by a derivation of the same or smaller size.*
- *If* $nonpos(A)$ *and* $nonneg(A)$ *and* $nonpos(B)$ *and* $nonneg(B)$ *and* $\Psi \vdash A \leq^{\pm} B$
  then $A = B$.

Proof. By a routine induction on the subtyping derivations. □

Lemma 7 (Transitivity of Declarative Subtyping)**.**
Given $\Psi \vdash A$ *type and* $\Psi \vdash B$ *type and* $\Psi \vdash C$ *type:*

(i) If $\mathcal{D}_1 :: \Psi \vdash A \leq^{\pm} B$ *and* $\mathcal{D}_2 :: \Psi \vdash B \leq^{\pm} C$
then $\Psi \vdash A \leq^{\pm} C$.

Proof. By lexicographic induction on (1) the sum of head quantifiers in A, B, and C, and (2) the size of the derivation.

We begin by case analysis on the shape of B, and the polarity of subtyping:

- Case $B = \forall\beta : \kappa_2.\, B'$, polarity $= -$:

  We case-analyze $\mathcal{D}_1$:

  - **Case** $\dfrac{\Psi \vdash \tau : \kappa_1 \qquad \Psi \vdash [\tau/\alpha]A' \leq^- B}{\Psi \vdash \forall\alpha : \kappa_1.\, A' \leq^- B} \leq\forall L$

$\Psi \vdash \tau : \kappa_1$	Subderivation
$\Psi \vdash [\tau/\alpha]A' \leq^- B$	Subderivation
$\Psi \vdash B \leq^- C$	Given
$\Psi \vdash [\tau/\alpha]A' \leq^- C$	By i.h. (A lost a quantifier)
$\Psi \vdash A \leq^- C$	By rule $\leq\forall L$

  - **Case** $\dfrac{\Psi, \beta : \kappa_2 \vdash A \leq^- B'}{\Psi \vdash A \leq^- \forall\beta : \kappa_2.\, B'} \leq\forall R$

    We case-analyze $\mathcal{D}_2$:

    * **Case** $\dfrac{\Psi \vdash \tau : \kappa_2 \qquad \Psi \vdash [\tau/\beta]B' \leq^- C}{\Psi \vdash \forall\beta : \kappa_2.\, B' \leq^- C} \leq\forall L$

$\Psi, \beta : \kappa_2 \vdash A \leq^- B'$	By Lemma 5 (Subtyping Inversion) on $\mathcal{D}_1$
$\Psi \vdash \tau : \kappa_2$	Subderivation
$\Psi \vdash [\tau/\beta]B' \leq^- C$	Subderivation of $\mathcal{D}_2$
$\Psi \vdash A \leq^- [\tau/\beta]B'$	By Lemma 3 (Declarative Term Substitution)
$\Psi \vdash A \leq^- C$	By i.h. (B lost a quantifier)

    * **Case** $\dfrac{\Psi, c : \kappa_3 \vdash B \leq^- C'}{\Psi \vdash B \leq^- \forall c : \kappa_3.\, C'} \leq\forall R$

$\Psi \vdash A \leq^- B$	Given
$\Psi, c : \kappa_3 \vdash A \leq^- B$	By Lemma 2 (Declarative Weakening)
$\Psi, c : \kappa_3 \vdash B \leq^- C'$	Subderivation
$\Psi, c : \kappa_3 \vdash A \leq^- C'$	By i.h. (C lost a quantifier)
$\Psi \vdash B \leq^- \forall c : \kappa_3.\, C'$	By $\leq\forall R$

- Case $nonpos(B)$, polarity $= +$:

  Now we case-analyze $\mathcal{D}_1$:

  - **Case** $\dfrac{\Psi, \alpha : \tau \vdash A' \leq^+ B}{\Psi \vdash \underbrace{\exists\alpha : \kappa_1.\, A'}_{A} \leq^+ B} \leq\exists L$

$\Psi, \alpha : \tau \vdash A' \leq^+ B$	Subderivation
$\Psi, \alpha : \tau \vdash B \leq^+ C$	By Lemma 2 (Declarative Weakening) ($\mathcal{D}_2$)
$\Psi, \alpha : \tau \vdash A' \leq^+ C$	By i.h. (A lost a quantifier)
$\Psi \vdash \exists\alpha : \kappa_1.\, A' \leq^+ C$	By $\leq\exists L$

  - **Case** $\dfrac{\Psi \vdash A \leq^- B \qquad nonpos(A) \qquad nonpos(B)}{\Psi \vdash A \leq^+ B} \leq^-_+$

    Now we case-analyze $\mathcal{D}_2$:

* **Case** $\dfrac{\Psi \vdash \tau : \kappa_3 \qquad \Psi \vdash B \leq^+ [\tau/c]C'}{\Psi \vdash B \leq^+ \underbrace{\exists c : \kappa_3 . C'}_{C}} \leq\exists R$

  $\Psi \vdash A \leq^+ B$ Given
  $\Psi \vdash \tau : \kappa_3$ Subderivation of $\mathcal{D}_2$
  $\Psi \vdash B \leq^+ [\tau/c]C'$ Subderivation of $\mathcal{D}_2$
  $\Psi \vdash A \leq^+ [\tau/c]C'$ By i.h. (C lost a quantifier)
  $\Psi \vdash A \leq^+ \exists c : \kappa_3 . C'$ By $\leq\exists R$

* **Case** $\dfrac{\Psi \vdash B \leq^- C \qquad nonpos(B) \qquad nonpos(C)}{\Psi \vdash B \leq^+ C} \leq^-_+$

  $\Psi \vdash A \leq^- B$ Subderivation of $\mathcal{D}_1$
  $\Psi \vdash B \leq^- C$ Subderivation of $\mathcal{D}_2$
  $\Psi \vdash A \leq^- C$ By i.h. ($\mathcal{D}_1$ and $\mathcal{D}_2$ smaller)
  $nonpos(A)$ Subderivation of $\mathcal{D}_1$
  $nonpos(C)$ Subderivation of $\mathcal{D}_2$
  $\Psi \vdash A \leq^+ C$ By $\leq^-_+$

- Case $B = \exists \beta : \kappa_2 . B'$, polarity $= +$:
  Now we case-analyze $\mathcal{D}_2$:

  – **Case** $\dfrac{\Psi \vdash \tau : \kappa_3 \qquad \Psi \vdash B \leq^+ [\tau/\alpha]C'}{\Psi \vdash B \leq^+ \underbrace{\exists \alpha : \kappa_3 . C'}_{C}} \leq\exists R$

  $\Psi \vdash \tau : \kappa_3$ Subderivation of $\mathcal{D}_2$
  $\Psi \vdash B \leq^+ [\tau/\alpha]C'$ Subderivation of $\mathcal{D}_2$
  $\Psi \vdash A \leq^+ B$ Given
  $\Psi \vdash A \leq^+ [\tau/\alpha]C'$ By i.h. (C lost a quantifier)
  $\Psi \vdash A \leq^+ C$ By rule $\leq\exists R$

  – **Case** $\dfrac{\Psi, \beta : \kappa_2 \vdash B' \leq^+ C}{\Psi \vdash \exists \beta : \kappa_2 . B' \leq^+ C} \leq\exists L$

  Now we case-analyze $\mathcal{D}_1$:

  * **Case** $\dfrac{\Psi \vdash \tau : \kappa_2 \qquad \Psi \vdash A \leq^+ [\tau/\beta]B'}{\Psi \vdash A \leq^+ \underbrace{\exists \beta : \kappa_2 . B'}_{B}} \leq\exists R$

  $\Psi, \beta : \kappa_2 \vdash B' \leq^+ C$ Subderivation of $\mathcal{D}_2$
  $\Psi \vdash \tau : \kappa_2$ Subderivation of $\mathcal{D}_1$
  $\Psi \vdash A \leq^+ [\tau/\beta]B'$ Subderivation of $\mathcal{D}_1$
  $\Psi \vdash [\tau/\beta]B' \leq^+ C$ By Lemma 3 (Declarative Term Substitution)
  $\Psi \vdash A \leq^+ C$ By i.h. (B lost a quantifier)

  * **Case** $\dfrac{\Psi, \alpha : \kappa_1 \vdash A \leq^+ B}{\Psi \vdash \underbrace{\exists \alpha : \kappa_1 . A'}_{A} \leq^+ B} \leq\exists L$

  $\Psi \vdash B \leq^+ C$ Given
  $\Psi, \alpha : \kappa_1 \vdash A' \leq^+ B$ Subderivation of $\mathcal{D}_1$
  $\Psi, \alpha : \kappa_1 \vdash A' \leq^+ B$ By Lemma 2 (Declarative Weakening)
  $\Psi, \alpha : \kappa_1 \vdash A' \leq^+ C$ By i.h. (A lost a quantifier)
  $\Psi \vdash \exists \alpha : \kappa_1 . A' \leq^+ C$ By $\leq\exists L$

- Case $\mathit{nonneg}(B)$, polarity $= -$:
  We case-analyze $\mathcal{D}_2$:

  - **Case** $\dfrac{\Psi, c : \kappa_3 \vdash B \leq^+ C'}{\Psi \vdash B \leq^+ \underbrace{\exists c : \kappa_3.\, C'}_{C}} \leq\forall R$

    $\Psi, c : \kappa_3 \vdash B \leq^+ C'$ Subderivation of $\mathcal{D}_2$
    $\Psi, c : \kappa_3 \vdash A \leq^+ B$ By Lemma 2 (Declarative Weakening)
    $\Psi, c : \kappa_3 \vdash A \leq^+ C'$ By i.h. (C lost a quantifier)
    $\Psi \vdash A \leq^+ \forall c : \kappa_3.\, C'$ By $\leq\forall R$

  - **Case** $\dfrac{\Psi \vdash B \leq^+ C \qquad \mathit{nonneg}(B) \qquad \mathit{nonneg}(C)}{\Psi \vdash B \leq^- C} \leq^+_-$

    We case-analyze $\mathcal{D}_1$:

    * **Case** $\dfrac{\Psi \vdash \tau : \kappa_1 \qquad \Psi \vdash [\tau/\alpha]A' \leq^- B}{\Psi \vdash \underbrace{\forall \alpha : \kappa_1.\, A'}_{A} \leq^- B} \leq\forall L$

      $\Psi \vdash B \leq^- C$ Given
      $\Psi \vdash \tau : \kappa_1$ Subderivation of $\mathcal{D}_1$
      $\Psi \vdash [\tau/\alpha]A' \leq^- B$ Subderivation of $\mathcal{D}_1$
      $\Psi \vdash [\tau/\alpha]A' \leq^- C$ By i.h. (A lost a quantifier)
      $\Psi \vdash \forall \alpha : \kappa_1.\, A' \leq^- C$ By $\leq\forall L$

    * **Case** $\dfrac{\Psi \vdash A \leq^+ B \qquad \mathit{nonpos}(A) \qquad \mathit{nonpos}(B)}{\Psi \vdash A \leq^- B} \leq^+_-$

      $\Psi \vdash A \leq^+ B$ Subderivation of $\mathcal{D}_1$
      $\Psi \vdash B \leq^+ C$ Subderivation of $\mathcal{D}_2$
      $\Psi \vdash A \leq^+ C$ By i.h. ($\mathcal{D}_1$ and $\mathcal{D}_2$ smaller)
      $\mathit{nonneg}(A)$ Subderivation of $\mathcal{D}_2$
      $\mathit{nonneg}(C)$ Subderivation of $\mathcal{D}_2$
      $\Psi \vdash A \leq^- C$ By $\leq^+_-$

□

C′ Substitution and Well-formedness Properties

Lemma 8 (Substitution—Well-formedness)**.**

(i) If $\Gamma \vdash A\ p$ *type and* $\Gamma \vdash \tau\ p$ *type then* $\Gamma \vdash [\tau/\alpha]A\ p$ *type.*

(ii) If $\Gamma \vdash P$ *prop and* $\Gamma \vdash \tau\ p$ *type then* $\Gamma \vdash [\tau/\alpha]P$ *prop.*
Moreover, if $p = !$ *and* $\mathsf{FEV}([\Gamma]P) = \emptyset$ *then* $\mathsf{FEV}([\Gamma][\tau/\alpha]P) = \emptyset$*.*

Proof. By induction on the derivations of $\Gamma \vdash A\ p$ *type* and $\Gamma \vdash P$ *prop*. □

Lemma 9 (Uvar Preservation)**.**
If $\Delta \longrightarrow \Omega$ *then:*

(i) If $(\alpha : \kappa) \in \Omega$ *then* $(\alpha : \kappa) \in [\Omega]\Delta$*.*

(ii) If $(x : A\ p) \in \Omega$ *then* $(x : [\Omega]A\ p) \in [\Omega]\Delta$*.*

Proof. By induction on Ω, following the definition of context application (Figure **??**). □

Lemma 10 (Sorting Implies Typing)**.** *If* $\Gamma \vdash t : \star$ *then* $\Gamma \vdash t$ *type.*

Proof. By induction on the given derivation. All cases are straightforward. □

Lemma 11 (Right-Hand Substitution for Sorting)**.** *If* $\Gamma \vdash t : \kappa$ *then* $\Gamma \vdash [\Gamma]t : \kappa$.

Proof. By induction on $|\Gamma \vdash t|$ (the size of t under Γ).

- **Cases** UnitSort: Here $t = 1$, so applying Γ to t does not change it: $t = [\Gamma]t$. Since $\Gamma \vdash t : \kappa$, we have $\Gamma \vdash [\Gamma]t : \kappa$, which was to be shown.
- **Case** VarSort: If t is an existential variable $\hat{\alpha}$, then $\Gamma = \Gamma_0[\hat{\alpha}]$, so applying Γ to t does not change it, and we proceed as in the UnitSort case above.

  If t is a universal variable α and Γ has no equation for it, then proceed as in the UnitSort case.

  Otherwise, $t = \alpha$ and $(\alpha = \tau) \in \Gamma$:

  $$\Gamma \quad = \quad (\Gamma_L, \alpha : \kappa, \Gamma_M, \alpha = \tau, \Gamma_R)$$

  By the implicit assumption that Γ is well-formed, $\Gamma_L, \alpha : \kappa, \Gamma_M \vdash \tau : \kappa$.
  By Lemma 34 (Suffix Weakening), $\Gamma \vdash \tau : \kappa$. Since $|\Gamma \vdash \tau| < |\Gamma \vdash \alpha|$, we can apply the i.h., giving

  $$\Gamma \vdash [\Gamma]\tau : \kappa$$

  By the definition of substitution, $[\Gamma]\tau = [\Gamma]\alpha$, so we have $\Gamma \vdash [\Gamma]\alpha : \kappa$.
- **Case** SolvedVarSort: In this case $t = \hat{\alpha}$ and $\Gamma = (\Gamma_L, \hat{\alpha} = \tau, \Gamma_R)$. Thus $[\Gamma]t = [\Gamma]\hat{\alpha} = [\Gamma_L]\tau$. We assume contexts are well-formed, so all free variables in τ are declared in Γ_L. Consequently, $|\Gamma_L \vdash \tau| = |\Gamma \vdash \tau|$, which is less than $|\Gamma \vdash \hat{\alpha}|$. We can therefore apply the i.h. to τ, yielding $\Gamma \vdash [\Gamma]\tau : \kappa$. By the definition of substitution, $[\Gamma]\tau = [\Gamma]\hat{\alpha}$, so we have $\Gamma \vdash [\Gamma]\hat{\alpha} : \kappa$.
- **Case** BinSort: In this case $t = t_1 \oplus t_2$. By i.h., $\Gamma \vdash [\Gamma]t_1 : \kappa$ and $\Gamma \vdash [\Gamma]t_2 : \kappa$. By BinSort, $\Gamma \vdash ([\Gamma]t_1) \oplus ([\Gamma]t_2) : \kappa$, which by the definition of substitution is $\Gamma \vdash [\Gamma](t_1 \oplus t_2) : \kappa$. □

Lemma 12 (Right-Hand Substitution for Propositions)**.** *If* $\Gamma \vdash P$ *prop then* $\Gamma \vdash [\Gamma]P$ *prop*.

Proof. Use inversion (EqProp), apply Lemma 11 (Right-Hand Substitution for Sorting) to each premise, and apply EqProp again. □

Lemma 13 (Right-Hand Substitution for Typing)**.** *If* $\Gamma \vdash A$ *type then* $\Gamma \vdash [\Gamma]A$ *type*.

Proof. By induction on $|\Gamma \vdash A|$ (the size of A under Γ).
Several cases correspond to cases in the proof of Lemma 11 (Right-Hand Substitution for Sorting):

- the case for UnitWF is like the case for UnitSort;
- the case for SolvedVarSort is like the cases for VarWF and SolvedVarWF,
- the case for VarSort is like the case for VarWF, but in the last subcase, apply Lemma 10 (Sorting Implies Typing) to move from a sorting judgment to a typing judgment.
- the case for BinWF is like the case for BinSort.

Now, the new cases:

- **Case** ForallWF: In this case $A = \forall\alpha : \kappa.\ A_0$. By i.h., $\Gamma, \alpha : \kappa \vdash [\Gamma, \alpha : \kappa]A_0$ *type*. By the definition of substitution, $[\Gamma, \alpha : \kappa]A_0 = [\Gamma]A_0$, so by ForallWF, $\Gamma \vdash \forall\alpha.\ [\Gamma]A_0$ *type*, which by the definition of substitution is $\Gamma \vdash [\Gamma](\forall\alpha.\ A_0)$ *type*.
- **Case** ExistsWF: Similar to the ForallWF case.
- **Case** ImpliesWF, WithWF: Use the i.h. and Lemma 12 (Right-Hand Substitution for Propositions), then apply ImpliesWF or WithWF. □

Lemma 14 (Substitution for Sorting)**.** *If* $\Omega \vdash t : \kappa$ *then* $[\Omega]\Omega \vdash [\Omega]t : \kappa$.

Proof. By induction on $|\Omega \vdash t|$ (the size of t under Ω).

- **Case** $\dfrac{u : \kappa \in \Omega}{\Omega \vdash u : \kappa}\ \mathsf{VarSort}$

  We have a complete context Ω, so u cannot be an existential variable: it must be some universal variable α.

  If Ω lacks an equation for α, use Lemma 9 (Uvar Preservation) and apply rule UvarSort.

  Otherwise, $(\alpha = \tau \in \Omega$, so we need to show $\Omega \vdash [\Omega]\tau : \kappa$. By the implicit assumption that Ω is well-formed, plus Lemma 34 (Suffix Weakening), $\Omega \vdash \tau : \kappa$. By Lemma 11 (Right-Hand Substitution for Sorting), $\Omega \vdash [\Omega]\tau : \kappa$.

- **Case** $\dfrac{\hat{\alpha} : \kappa = \tau \in \Omega}{\Omega \vdash \hat{\alpha} : \kappa}\ \mathsf{SolvedVarSort}$

$\hat{\alpha} : \kappa = \tau \in \Omega$	Subderivation
$\Omega = (\Omega_L, \hat{\alpha} : \kappa = \tau, \Omega_R)$	Decomposing Ω
$\Omega_L \vdash \tau : \kappa$	By implicit assumption that Ω is well-formed
$\Omega_L, \hat{\alpha} : \kappa = \tau, \Omega_R \vdash \tau : \kappa$	By Lemma 34 (Suffix Weakening)
$\Omega \vdash [\Omega]\tau : \kappa$	By Lemma 11 (Right-Hand Substitution for Sorting)
☞ $[\Omega]\Omega \vdash [\Omega]\hat{\alpha} : \kappa$	$[\Omega]\tau = [\Omega]\hat{\alpha}$

- **Case** $\dfrac{}{\Omega \vdash 1 : \star}\ \mathsf{UnitSort}$

  Since $1 = [\Omega]1$, applying UnitSort gives the result.

- **Case** $\dfrac{\Omega \vdash \tau_1 : \star \qquad \Omega \vdash \tau_2 : \star}{\Omega \vdash \tau_1 \oplus \tau_2 : \star}\ \mathsf{BinSort}$

  By i.h. on each premise, rule BinSort, and the definition of substitution.

- **Case** $\dfrac{}{\Omega \vdash \mathsf{zero} : \mathbb{N}}\ \mathsf{ZeroSort}$

  Since $\mathsf{zero} = [\Omega]\mathsf{zero}$, applying ZeroSort gives the result.

- **Case** $\dfrac{\Omega \vdash t : \mathbb{N}}{\Omega \vdash \mathsf{succ}(t) : \mathbb{N}}\ \mathsf{SuccSort}$

  By i.h., rule SuccSort, and the definition of substitution. □

Lemma 15 (Substitution for Prop Well-Formedness)**.**
If $\Omega \vdash P$ *prop then* $[\Omega]\Omega \vdash [\Omega]P$ *prop.*

Proof. Only one rule derives this judgment form:

- **Case** $\dfrac{\Omega \vdash t : \mathbb{N} \qquad \Omega \vdash t' : \mathbb{N}}{\Omega \vdash t = t'\ \mathit{prop}}\ \mathsf{EqProp}$

$\Omega \vdash t : \mathbb{N}$	Subderivation
$[\Omega]\Omega \vdash [\Omega]t : \mathbb{N}$	By Lemma 14 (Substitution for Sorting)
$\Omega \vdash t' : \mathbb{N}$	Subderivation
$[\Omega]\Omega \vdash [\Omega]t' : \mathbb{N}$	By Lemma 14 (Substitution for Sorting)
$[\Omega]\Omega \vdash ([\Omega]t) = ([\Omega]t')$ *prop*	By EqProp
☞ $[\Omega]\Omega \vdash [\Omega](t = t')$ *prop*	By def. of subst.

  □

Lemma 16 (Substitution for Type Well-Formedness)**.** *If* $\Omega \vdash A$ *type then* $[\Omega]\Omega \vdash [\Omega]A$ *type.*

Proof. By induction on $|\Omega \vdash A|$.

Several cases correspond to those in the proof of Lemma 14 (Substitution for Sorting):

- the UnitWF case is like the UnitSort case (using DeclUnitWF instead of UnitSort);
- the VarWF case is like the VarSort case (using DeclUvarWF instead of UvarSort);
- the SolvedVarWF case is like the SolvedVarSort case.

However, uses of Lemma 11 (Right-Hand Substitution for Sorting) are replaced by uses of Lemma 13 (Right-Hand Substitution for Typing).

Now, the new cases:

- **Case** $\dfrac{\Omega, \alpha : \kappa \vdash A_0 \; \mathit{type}}{\Omega \vdash \forall \alpha : \kappa.\, A_0 \; \mathit{type}}$ ForallWF

$$
\begin{array}{rll}
\Omega, \alpha : \kappa \vdash A_0 : \kappa' & & \text{Subderivation} \\
[\Omega, \alpha : \kappa](\Omega, \alpha : \kappa) \vdash [\Omega]A_0 : \kappa' & & \text{By i.h.} \\
[\Omega]\Omega, \alpha : \kappa \vdash [\Omega]A_0 : \kappa' & & \text{By definition of completion} \\
[\Omega]\Omega \vdash \forall \alpha : \kappa.\, [\Omega]A_0 : \kappa' & & \text{By DeclAllWF} \\
\text{☞} \quad [\Omega]\Omega \vdash [\Omega](\forall \alpha : \kappa.\, A_0) : \kappa' & & \text{By def. of subst.}
\end{array}
$$

- **Case** ExistsWF: Similar to the ForallWF case, using DeclExistsWF instead of DeclAllWF.

- **Case** $\dfrac{\Omega \vdash A_1 \; \mathit{type} \qquad \Omega \vdash A_2 \; \mathit{type}}{\Omega \vdash A_1 \oplus A_2 \; \mathit{type}}$ BinWF

  By i.h. on each premise, rule DeclBinWF, and the definition of substitution.

- **Case** VecWF: Similar to the BinWF case.

- **Case** $\dfrac{\Omega \vdash \mathsf{P} \; \mathit{prop} \qquad \Omega \vdash A_0 \; \mathit{type}}{\Omega \vdash \mathsf{P} \supset A_0 \; \mathit{type}}$ ImpliesWF

$$
\begin{array}{rll}
\Omega \vdash \mathsf{P} \; \mathit{prop} & & \text{Subderivation} \\
[\Omega]\Omega \vdash [\Omega]\mathsf{P} \; \mathit{prop} & & \text{By Lemma 15 (Substitution for Prop Well-Formedness)} \\
\Omega \vdash A_0 \; \mathit{type} & & \text{Subderivation} \\
[\Omega]\Omega \vdash [\Omega]A_0 \; \mathit{type} & & \text{By i.h.} \\
[\Omega]\Omega \vdash ([\Omega]\mathsf{P}) \supset ([\Omega]A_0) \; \mathit{type} & & \text{By DeclImpliesWF} \\
\text{☞} \quad [\Omega]\Omega \vdash [\Omega](\mathsf{P} \supset A_0) \; \mathit{type} & & \text{By def. of subst.}
\end{array}
$$

- **Case** $\dfrac{\Omega \vdash \mathsf{P} \; \mathit{prop} \qquad \Omega \vdash A_0 \; \mathit{type}}{\Omega \vdash A_0 \wedge \mathsf{P} \; \mathit{type}}$ WithWF

  Similar to the ImpliesWF case. □

Lemma 17 (Substitution Stability)**.**
If (Ω, Ω_Z) *is well-formed and* Ω_Z *is soft and* $\Omega \vdash A \; \mathit{type}$ *then* $[\Omega]A = [\Omega, \Omega_Z]A$.

Proof. By induction on Ω_Z.

Since Ω_Z is soft, either (1) $\Omega_Z = \cdot$ (and the result is immediate) or (2) $\Omega_Z = (\Omega', \hat{\alpha} : \kappa)$ or (3) $\Omega_Z = (\Omega', \hat{\alpha} : \kappa = t)$. However, according to the grammar for complete contexts such as Ω_Z, (2) is impossible. Only case (3) remains.

By i.h., $[\Omega]A = [\Omega, \Omega']A$. Use the fact that $\Omega \vdash A \; \mathit{type}$ implies $\mathrm{FV}(A) \cap \mathsf{dom}(\Omega_Z) = \emptyset$. □

Lemma 18 (Equal Domains)**.**
If $\Omega_1 \vdash A \; \mathit{type}$ *and* $\mathsf{dom}(\Omega_1) = \mathsf{dom}(\Omega_2)$ *then* $\Omega_2 \vdash A \; \mathit{type}$.

Proof. By induction on the given derivation. □

D′ Properties of Extension

Lemma 19 (Declaration Preservation)**.** *If* $\Gamma \longrightarrow \Delta$ *and* u *is declared in* Γ*, then* u *is declared in* Δ*.*

Proof. By induction on the derivation of $\Gamma \longrightarrow \Delta$.

- **Case**
$$\frac{}{\cdot \longrightarrow \cdot}\ \longrightarrow\mathsf{Id}$$
This case is impossible, since by hypothesis u is declared in Γ.

- **Case** $\dfrac{\Gamma \longrightarrow \Delta \qquad [\Delta]A = [\Delta]A'}{\Gamma, x : A \longrightarrow \Delta, x : A'}\ \longrightarrow\mathsf{Var}$
  - Case $u = x$: Immediate.
  - Case $u \neq x$: Since u is declared in $(\Gamma, x : A)$, it is declared in Γ. By i.h., u is declared in Δ, and therefore declared in $(\Delta, x : A')$.

- **Case** $\dfrac{\Gamma \longrightarrow \Delta}{\Gamma, \alpha : \kappa \longrightarrow \Delta, \alpha : \kappa}\ \longrightarrow\mathsf{Uvar}$
Similar to the $\longrightarrow$Var case.

- **Case** $\dfrac{\Gamma \longrightarrow \Delta}{\Gamma, \hat{\alpha} : \kappa \longrightarrow \Delta, \hat{\alpha} : \kappa}\ \longrightarrow\mathsf{Unsolved}$
Similar to the $\longrightarrow$Var case.

- **Case** $\dfrac{\Gamma \longrightarrow \Delta \qquad [\Delta]t = [\Delta]t'}{\Gamma, \hat{\alpha} : \kappa = t \longrightarrow \Delta, \hat{\alpha} : \kappa = t'}\ \longrightarrow\mathsf{Solved}$
Similar to the $\longrightarrow$Var case.

- **Case** $\dfrac{\Gamma \longrightarrow \Delta \qquad [\Delta]t = [\Delta]t'}{\Gamma, \alpha = t \longrightarrow \Delta, \alpha = t'}\ \longrightarrow\mathsf{Eqn}$
It is given that u is declared in $(\Gamma, \alpha = t)$. Since $\alpha = t$ is not a declaration, u is declared in Γ. By i.h., u is declared in Δ, and therefore declared in $(\Delta, \alpha = t'$.

- **Case** $\dfrac{\Gamma \longrightarrow \Delta}{\Gamma, \blacktriangleright_{\hat{\alpha}} \longrightarrow \Delta, \blacktriangleright_{\hat{\alpha}}}\ \longrightarrow\mathsf{Marker}$
Similar to the $\longrightarrow$Eqn case.

- **Case** $\dfrac{\Gamma \longrightarrow \Delta}{\Gamma, \hat{\beta} : \kappa' \longrightarrow \Delta, \hat{\beta} : \kappa' = t}\ \longrightarrow\mathsf{Solve}$
Similar to the $\longrightarrow$Var case.

- **Case** $\dfrac{\Gamma \longrightarrow \Delta}{\Gamma \longrightarrow \Delta, \hat{\alpha} : \kappa}\ \longrightarrow\mathsf{Add}$
It is given that u is declared in Γ. By i.h., u is declared in Δ, and therefore declared in $(\Delta, \hat{\alpha} : \kappa)$.

- **Case** $\dfrac{\Gamma \longrightarrow \Delta}{\Gamma \longrightarrow \Delta, \hat{\alpha} : \kappa = t}\ \longrightarrow\mathsf{AddSolved}$
Similar to the $\longrightarrow$Add case. □

Lemma 20 (Declaration Order Preservation)**.** *If* $\Gamma \longrightarrow \Delta$ *and* u *is declared to the left of* v *in* Γ*, then* u *is declared to the left of* v *in* Δ*.*

Proof. By induction on the derivation of $\Gamma \longrightarrow \Delta$.

- **Case**
$$\frac{}{\cdot \longrightarrow \cdot}\ \longrightarrow\mathsf{Id}$$
This case is impossible, since by hypothesis u and v are declared in Γ.

- **Case** $\dfrac{\Gamma \longrightarrow \Delta \qquad [\Delta]A = [\Delta]A'}{\Gamma, x : A \longrightarrow \Delta, x : A'}\ \longrightarrow\mathsf{Var}$
Consider whether $v = x$:
  - Case $v = x$:
    It is given that u is declared to the left of v in $(\Gamma, x : A)$, so u is declared in Γ.
    By Lemma 19 (Declaration Preservation), u is declared in Δ.
    Therefore u is declared to the left of v in $(\Delta, x : A')$.
  - Case $v \neq x$:
    Here, v is declared in Γ. By i.h., u is declared to the left of v in Δ.
    Therefore u is declared to the left of v in $(\Delta, x : A')$.

- **Case** $\dfrac{\Gamma \longrightarrow \Delta}{\Gamma, \alpha : \kappa \longrightarrow \Delta, \alpha : \kappa}\ \longrightarrow\mathsf{Uvar}$
Similar to the $\longrightarrow$Var case.

- **Case** $\dfrac{\Gamma \longrightarrow \Delta}{\Gamma, \hat{\alpha} : \kappa \longrightarrow \Delta, \hat{\alpha} : \kappa}\ \longrightarrow\mathsf{Unsolved}$
Similar to the $\longrightarrow$Var case.

- **Case** $\dfrac{\Gamma \longrightarrow \Delta \qquad [\Delta]t = [\Delta]t'}{\Gamma, \hat{\alpha} : \kappa = t \longrightarrow \Delta, \hat{\alpha} : \kappa = t'}\ \longrightarrow\mathsf{Solved}$
Similar to the $\longrightarrow$Var case.

- **Case** $\dfrac{\Gamma \longrightarrow \Delta}{\Gamma, \hat{\beta} : \kappa' \longrightarrow \Delta, \hat{\beta} : \kappa' = t}\ \longrightarrow\mathsf{Solve}$
Similar to the $\longrightarrow$Var case.

- **Case** $\dfrac{\Gamma \longrightarrow \Delta \qquad [\Delta]t = [\Delta]t'}{\Gamma, \alpha = t \longrightarrow \Delta, \alpha = t'}\ \longrightarrow\mathsf{Eqn}$
The equation $\hat{\alpha} = t$ does not declare any variables, so u and v must be declared in Γ.
By i.h., u is declared to the left of v in Δ.
Therefore u is declared to the left of v in $\Delta, \hat{\alpha} : \kappa = t'$.

- **Case** $\dfrac{\Gamma \longrightarrow \Delta}{\Gamma, \blacktriangleright_{\hat{\alpha}} \longrightarrow \Delta, \blacktriangleright_{\hat{\alpha}}}\ \longrightarrow\mathsf{Marker}$
Similar to the $\longrightarrow$Eqn case.

- **Case** $\dfrac{\Gamma \longrightarrow \Delta}{\Gamma \longrightarrow \Delta, \hat{\alpha} : \kappa}\ \longrightarrow\mathsf{Add}$

  By i.h., u is declared to the left of v in Δ.
  Therefore u is declared to the left of v in $(\Delta, \hat{\alpha} : \kappa)$.

- **Case** $\dfrac{\Gamma \longrightarrow \Delta}{\Gamma \longrightarrow \Delta, \hat{\alpha} : \kappa = t}\ \longrightarrow\mathsf{AddSolved}$

  Similar to the $\longrightarrow$Add case. □

Lemma 21 (Reverse Declaration Order Preservation)**.** *If $\Gamma \longrightarrow \Delta$ and u and v are both declared in Γ and u is declared to the left of v in Δ, then u is declared to the left of v in Γ.*

Proof. It is given that u and v are declared in Γ. Either u is declared to the left of v in Γ, or v is declared to the left of u. Suppose the latter (for a contradiction). By Lemma 20 (Declaration Order Preservation), v is declared to the left of u in Δ. But we know that u is declared to the left of v in Δ: contradiction. Therefore u is declared to the left of v in Γ. □

Lemma 22 (Extension Inversion)**.**

(i) *If $\mathcal{D} :: \Gamma_0, \alpha : \kappa, \Gamma_1 \longrightarrow \Delta$*
then there exist unique Δ_0 and Δ_1
such that $\Delta = (\Delta_0, \alpha : \kappa, \Delta_1)$ and $\mathcal{D}' :: \Gamma_0 \longrightarrow \Delta_0$ where $\mathcal{D}' < \mathcal{D}$.
Moreover, if Γ_1 is soft, then Δ_1 is soft.

(ii) *If $\mathcal{D} :: \Gamma_0, \blacktriangleright_u, \Gamma_1 \longrightarrow \Delta$*
then there exist unique Δ_0 and Δ_1
such that $\Delta = (\Delta_0, \blacktriangleright_u, \Delta_1)$ and $\mathcal{D}' :: \Gamma_0 \longrightarrow \Delta_0$ where $\mathcal{D}' < \mathcal{D}$.
Moreover, if Γ_1 is soft, then Δ_1 is soft.
Moreover, if $\mathsf{dom}(\Gamma_0, \blacktriangleright_u, \Gamma_1) = \mathsf{dom}(\Delta)$ then $\mathsf{dom}(\Gamma_0) = \mathsf{dom}(\Delta_0)$.

(iii) *If $\mathcal{D} :: \Gamma_0, \alpha = \tau, \Gamma_1 \longrightarrow \Delta$*
then there exist unique Δ_0, τ', and Δ_1
such that $\Delta = (\Delta_0, \alpha = \tau', \Delta_1)$ and $\mathcal{D}' :: \Gamma_0 \longrightarrow \Delta_0$ and $[\Delta_0]\tau = [\Delta_0]\tau'$ where $\mathcal{D}' < \mathcal{D}$.

(iv) *If $\mathcal{D} :: \Gamma_0, \hat{\alpha} : \kappa = \tau, \Gamma_1 \longrightarrow \Delta$*
then there exist unique Δ_0, τ', and Δ_1
such that $\Delta = (\Delta_0, \hat{\alpha} : \kappa = \tau', \Delta_1)$ and $\mathcal{D}' :: \Gamma_0 \longrightarrow \Delta_0$ and $[\Delta_0]\tau = [\Delta_0]\tau'$ where $\mathcal{D}' < \mathcal{D}$.

(v) *If $\mathcal{D} :: \Gamma_0, x : A, \Gamma_1 \longrightarrow \Delta$*
then there exist unique Δ_0, A', and Δ_1
such that $\Delta = (\Delta_0, x : A', \Delta_1)$ and $\mathcal{D}' :: \Gamma_0 \longrightarrow \Delta_0$ and $[\Delta_0]A = [\Delta_0]A'$ where $\mathcal{D}' < \mathcal{D}$.
Moreover, if Γ_1 is soft, then Δ_1 is soft.
Moreover, if $\mathsf{dom}(\Gamma_0, x : A, \Gamma_1) = \mathsf{dom}(\Delta)$ then $\mathsf{dom}(\Gamma_0) = \mathsf{dom}(\Delta_0)$.

(vi) *If $\mathcal{D} :: \Gamma_0, \hat{\alpha} : \kappa, \Gamma_1 \longrightarrow \Delta$ then either*

- *there exist unique Δ_0, τ', and Δ_1*
  such that $\Delta = (\Delta_0, \hat{\alpha} : \kappa = \tau', \Delta_1)$ and $\mathcal{D}' :: \Gamma_0 \longrightarrow \Delta_0$ where $\mathcal{D}' < \mathcal{D}$,
  or
- *there exist unique Δ_0 and Δ_1*
  such that $\Delta = (\Delta_0, \hat{\alpha} : \kappa, \Delta_1)$ and $\mathcal{D}' :: \Gamma_0 \longrightarrow \Delta_0$ where $\mathcal{D}' < \mathcal{D}$.

Proof. In each part, we proceed by induction on the derivation of $\Gamma_0, \ldots, \Gamma_1 \longrightarrow \Delta$.
Note that in each part, the $\longrightarrow$Id case is impossible.
Throughout this proof, we shadow Δ so that it refers to the *largest proper prefix* of the Δ in the statement of the lemma. For example, in the $\longrightarrow$Var case of part (i), we really have $\Delta = (\Delta_{00}, x : A')$, but we call Δ_{00} "Δ".

(i) We have $\Gamma_0, \alpha : \kappa, \Gamma_1 \longrightarrow \Delta$.

- **Case** $\dfrac{\Gamma \longrightarrow \Delta \qquad [\Delta]A = [\Delta]A'}{\underbrace{\Gamma, x : A}_{\Gamma_0, \alpha:\kappa, \Gamma_1} \longrightarrow \Delta, x : A'} \longrightarrow\mathsf{Var}$

$$
\begin{array}{rll}
(\Gamma, x : A) = (\Gamma_0, \alpha : \kappa, \Gamma_1) & & \text{Given} \\
= (\Gamma_0, \alpha : \kappa, \Gamma_1', x : A) & & \text{Since the last element must be equal} \\
(\Gamma, x : A) = (\Gamma_0, \alpha : \kappa, \Gamma_1', x : A) & & \text{By transitivity} \\
\Gamma = (\Gamma_0, \alpha : \kappa, \Gamma_1') & & \text{By injectivity of syntax} \\
\\
\Gamma \longrightarrow \Delta & & \text{Subderivation} \\
\Gamma_0, \alpha : \kappa, \Gamma_1' \longrightarrow \Delta & & \text{By equality} \\
\Delta = (\Delta_0, \alpha : \kappa, \Delta_1) & & \text{By i.h.} \\
\text{☞} \quad \Gamma_0 \longrightarrow \Delta_0 & & '' \\
\text{if } \Gamma_1' \text{ soft then } \Delta_1 \text{ soft} & & '' \\
\\
\text{☞} \quad (\Delta, x : A') = (\Delta_0, \alpha : \kappa, \Delta_1, x : A') & & \text{By congruence} \\
\text{☞} \quad \text{if } \Gamma_1', x : A \text{ soft then } \Delta_1, x : A' \text{ soft} & & \text{Since } \Gamma_1', x : A \text{ is not soft}
\end{array}
$$

- **Case** $\dfrac{\Gamma \longrightarrow \Delta}{\underbrace{\Gamma, \beta : \kappa'}_{\Gamma_0, \alpha:\kappa, \Gamma_1} \longrightarrow \Delta, \beta : \kappa'} \longrightarrow\mathsf{Uvar}$

  There are two cases:

  - **Case** $\alpha : \kappa = \beta : \kappa'$:

$$
\begin{array}{rll}
\text{☞} \quad (\Gamma, \alpha : \kappa) = (\Gamma_0, \alpha : \kappa, \Gamma_1) & & \text{where } \Gamma_0 = \Gamma \text{ and } \Gamma_1 = \cdot \\
\text{☞} \quad (\Delta, \alpha : \kappa) = (\Delta_0, \alpha : \kappa, \Delta_1) & & \text{where } \Delta_0 = \Delta \text{ and } \Delta_1 = \cdot \\
\text{☞} \quad \text{if } \Gamma_1 \text{ soft then } \Delta_1 \text{ soft} & & \text{since } \cdot \text{ is soft}
\end{array}
$$

  - **Case** $\alpha \neq \beta$:

$$
\begin{array}{rll}
(\Gamma, \beta : \kappa') = (\Gamma_0, \alpha : \kappa, \Gamma_1) & & \text{Given} \\
= (\Gamma_0, \alpha : \kappa, \Gamma_1', \beta : \kappa') & & \text{Since the last element must be equal} \\
\Gamma = (\Gamma_0, \alpha : \kappa, \Gamma_1') & & \text{By injectivity of syntax} \\
\\
\Gamma \longrightarrow \Delta & & \text{Subderivation} \\
\Gamma_0, \alpha : \kappa, \Gamma_1' \longrightarrow \Delta & & \text{By equality} \\
\Delta = (\Delta_0, \alpha : \kappa, \Delta_1) & & \text{By i.h.} \\
\text{☞} \quad \Gamma_0 \longrightarrow \Delta_0 & & '' \\
\text{if } \Gamma_1' \text{ soft then } \Delta_1 \text{ soft} & & '' \\
\\
\text{☞} \quad (\Delta, \beta : \kappa') = (\Delta_0, \alpha : \kappa, \Delta_1, \beta : \kappa') & & \text{By congruence} \\
\text{☞} \quad \text{if } \Gamma_1', \beta : \kappa' \text{ soft then } \Delta_1, \beta : \kappa' \text{ soft} & & \text{Since } \Gamma_1', \beta : \kappa' \text{ is not soft}
\end{array}
$$

- **Case** $\dfrac{\Gamma \longrightarrow \Delta}{\underbrace{\Gamma, \widehat{\alpha} : \kappa'}_{\Gamma_0, \alpha:\kappa, \Gamma_1} \longrightarrow \Delta, \widehat{\alpha} : \kappa'} \longrightarrow\mathsf{Unsolved}$

$$
\begin{array}{rl}
(\Gamma, \hat{\alpha} : \kappa') = (\Gamma_0, \alpha : \kappa, \Gamma_1) & \text{Given} \\
= (\Gamma_0, \alpha : \kappa, \Gamma_1', \hat{\alpha} : \kappa') & \text{Since the last element must be equal} \\
\Gamma = (\Gamma_0, \alpha : \kappa, \Gamma_1') & \text{By injectivity of syntax}
\end{array}
$$

$$
\begin{array}{rl}
\Gamma \longrightarrow \Delta & \text{Subderivation} \\
\Gamma_0, \alpha : \kappa, \Gamma_1' \longrightarrow \Delta & \text{By equality} \\
\Delta = (\Delta_0, \alpha : \kappa, \Delta_1) & \text{By i.h.} \\
\Gamma_0 \longrightarrow \Delta_0 & \text{☞ } '' \\
\text{if } \Gamma_1' \text{ soft then } \Delta_1 \text{ soft} & \text{☞ } ''
\end{array}
$$

☞ $(\Delta, \hat{\alpha} : \kappa') = (\Delta_0, \alpha : \kappa, \Delta_1, \hat{\alpha} : \kappa')$ By congruence

Suppose $\Gamma_1', \hat{\alpha} : \kappa'$ soft.

$$
\begin{array}{rl}
\Gamma_1' \text{ soft} & \text{By definition of softness} \\
\Delta_1 \text{ soft} & \text{By induction} \\
\Delta_1 \text{ soft} & \text{By definition of softness} \\
\text{☞ if } \Gamma_1', \hat{\alpha} : \kappa' \text{ soft then } \Delta_1, \hat{\alpha} : \kappa' \text{ soft} & \text{Implication introduction}
\end{array}
$$

- **Case** $\dfrac{\Gamma \longrightarrow \Delta \qquad [\Delta]t = [\Delta]t'}{\underbrace{\Gamma, \hat{\alpha} : \kappa = t}_{\Gamma_0, \alpha:\kappa, \Gamma_1} \longrightarrow \Delta, \hat{\alpha} : \kappa = t'} \longrightarrow\text{Solved}$

  Similar to the $\longrightarrow$Unsolved case.

- **Case** $\dfrac{\Gamma \longrightarrow \Delta \qquad [\Delta]t = [\Delta]t'}{\underbrace{\Gamma, \beta = t}_{\Gamma_0, \alpha:\kappa, \Gamma_1} \longrightarrow \Delta, \beta = t'} \longrightarrow\text{Eqn}$

$$
\begin{array}{rl}
(\Gamma, \beta = t) = (\Gamma_0, \alpha : \kappa, \Gamma_1) & \text{Given} \\
= (\Gamma_0, \alpha : \kappa, \Gamma_1', \beta = t) & \text{Since the last element must be equal} \\
\Gamma = (\Gamma_0, \alpha : \kappa, \Gamma_1') & \text{By injectivity of syntax}
\end{array}
$$

$$
\begin{array}{rl}
\Gamma \longrightarrow \Delta & \text{Subderivation} \\
\Gamma_0, \alpha : \kappa, \Gamma_1' \longrightarrow \Delta & \text{By equality} \\
\Delta = (\Delta_0, \alpha : \kappa, \Delta_1) & \text{By i.h.} \\
\Gamma_0 \longrightarrow \Delta_0 & \text{☞ } '' \\
\text{if } \Gamma_1' \text{ soft then } \Delta_1 \text{ soft} & \text{☞ } ''
\end{array}
$$

☞ $(\Delta, \beta = t') = (\Delta_0, \alpha : \kappa, \Delta_1, \beta = t')$ By congruence

☞ if $\Gamma_1', \beta = t$ soft then $\Delta_1, \beta = t'$ soft Since $\Gamma_1', \beta = t$ is not soft

- **Case** $\dfrac{\Gamma \longrightarrow \Delta}{\underbrace{\Gamma, \blacktriangleright\hat{\alpha}}_{\Gamma_0, \alpha:\kappa, \Gamma_1} \longrightarrow \Delta, \blacktriangleright\hat{\alpha}} \longrightarrow\text{Marker}$

$$
\begin{array}{rl}
(\Gamma, \blacktriangleright\hat{\alpha}) = (\Gamma_0, \alpha : \kappa, \Gamma_1) & \text{Given} \\
= (\Gamma_0, \alpha : \kappa, \Gamma_1', \blacktriangleright\hat{\alpha}) & \text{Since the last element must be equal} \\
\Gamma = (\Gamma_0, \alpha : \kappa, \Gamma_1') & \text{By injectivity of syntax}
\end{array}
$$

$$
\begin{array}{rl}
\Gamma \longrightarrow \Delta & \text{Subderivation} \\
\Gamma_0, \alpha : \kappa, \Gamma_1' \longrightarrow \Delta & \text{By equality} \\
\Delta = (\Delta_0, \alpha : \kappa, \Delta_1) & \text{By i.h.} \\
\Gamma_0 \longrightarrow \Delta_0 & \text{☞ } '' \\
\text{if } \Gamma_1' \text{ soft then } \Delta_1 \text{ soft} & \text{☞ } ''
\end{array}
$$

☞ $\Delta, \blacktriangleright\hat{\alpha} = (\Delta_0, \alpha : \kappa, \Delta_1, \blacktriangleright\hat{\alpha})$ By congruence

☞ if $\Gamma_1', \blacktriangleright\hat{\alpha}$ soft then $\Delta_1, \blacktriangleright\hat{\alpha}$ soft Since $\Gamma_1', \blacktriangleright\hat{\alpha}$ is not soft

- **Case** $\dfrac{\Gamma \longrightarrow \Delta}{\underbrace{\Gamma}_{\Gamma_0, \alpha:\kappa', \Gamma_1} \longrightarrow \Delta, \hat{\alpha} : \kappa} \longrightarrow\mathsf{Add}$

	$\Delta = (\Delta_0, \alpha : \kappa, \Delta_1)$	By i.h.
☞	$\Gamma_0 \longrightarrow \Delta_0$	$''$
	if Γ_1 soft then Δ_1 soft	$''$
☞	$\Delta, \hat{\alpha} : \kappa' = (\Delta_0, \alpha : \kappa, \Delta_1, \hat{\alpha} : \kappa')$	By congruence of equality

Suppose Γ_1 soft.

	Δ_1 soft	By i.h.
	$\Delta_1, \hat{\alpha} : \kappa'$ soft	By definition of softnesss
☞	if Γ_1 soft then $\Delta_1, \hat{\alpha} : \kappa'$ soft	Implication introduction

- **Case** $\dfrac{\Gamma \longrightarrow \Delta}{\underbrace{\Gamma}_{\Gamma_0, \alpha:\kappa, \Gamma_1} \longrightarrow \Delta, \hat{\alpha} : \kappa' = t} \longrightarrow\mathsf{AddSolved}$

	$\Delta = (\Delta_0, \alpha : \kappa, \Delta_1)$	By i.h.
☞	$\Gamma_0 \longrightarrow \Delta_0$	$''$
	if Γ_1 soft then Δ_1 soft	$''$
☞	$(\Delta, \hat{\alpha} : \kappa' = t) = (\Delta_0, \alpha : \kappa, \Delta_1, \hat{\alpha} : \kappa' = t)$	By congruence of equality

Suppose Γ_1 soft.

	Δ_1 soft	By i.h.
	$(\Delta_1, \hat{\alpha} : \kappa' = t)$ soft	By definition of softnesss
☞	if Γ_1 soft then $\Delta_1, \hat{\alpha} : \kappa' = t$ soft	Implication introduction

- **Case** $\dfrac{\Gamma \longrightarrow \Delta}{\underbrace{\Gamma, \hat{\beta} : \kappa'}_{\Gamma_0, \alpha:\kappa, \Gamma_1} \longrightarrow \Delta, \hat{\beta} : \kappa' = t} \longrightarrow\mathsf{Solve}$

	$(\Gamma, \hat{\beta} : \kappa') = (\Gamma_0, \alpha : \kappa, \Gamma_1)$	Given
	$= (\Gamma_0, \alpha : \kappa, \Gamma_1', \hat{\beta} : \kappa')$	Since the final elements are equal
	$\Gamma = (\Gamma_0, \alpha : \kappa, \Gamma_1')$	By injectivity of context syntax
	$\Gamma \longrightarrow \Delta$	Subderivation
	$\Gamma_0, \alpha : \kappa, \Gamma_1' \longrightarrow \Delta$	By equality
	$\Delta = (\Delta_0, \alpha : \kappa, \Delta_1)$	By i.h.
☞	$\Gamma_0 \longrightarrow \Delta_0$	$''$
	if Γ_1' soft then Δ_1 soft	$''$
☞	$\Delta, \hat{\beta} : \kappa' = \Delta_0, \alpha : \kappa, \Delta_1, \hat{\beta} : \kappa'$	By congruence

Suppose $\Gamma_1', \hat{\beta} : \kappa'$ soft.

	Γ_1' soft	By definition of softness
	Δ_1 soft	Using i.h.
	$\Delta_1, \hat{\beta} : \kappa' = t$ soft	By definition of softness
☞	if $\Gamma_1', \hat{\beta} : \kappa'$ soft then $\Delta_1, \hat{\beta} : \kappa' = t$ soft	Implication intro

(ii) We have $\Gamma_0, \blacktriangleright_u, \Gamma_1 \longrightarrow \Delta$. This part is similar to part (i) above, except for "if $\mathsf{dom}(\Gamma_0, \blacktriangleright_u, \Gamma_1) = \mathsf{dom}(\Delta)$ then $\mathsf{dom}(\Gamma_0) = \mathsf{dom}(\Delta_0)$", which follows by i.h. in most cases. In the $\longrightarrow$Marker case, either we have $\ldots, \blacktriangleright_{u'}$ where $u' = u$—in which case the i.h. gives us what we need—or we have a matching $\blacktriangleright_u$. In this latter case, we have $\Gamma_1 = \cdot$. We know that $\mathsf{dom}(\Gamma_0, \blacktriangleright_u, \Gamma_1) = \mathsf{dom}(\Delta)$ and $\Delta = (\Delta_0, \blacktriangleright_u)$. Since $\Gamma_1 = \cdot$, we have $\mathsf{dom}(\Gamma_0, \blacktriangleright_u) = \mathsf{dom}(\Delta_0, \blacktriangleright_u)$. Therefore $\mathsf{dom}(\Gamma_0) = \mathsf{dom}(\Delta_0)$.

(iii) We have $\Gamma_0, \alpha = \tau, \Gamma_1 \longrightarrow \Delta$.

- **Case** $\dfrac{\Gamma \longrightarrow \Delta}{\underbrace{\Gamma, \beta : \kappa'}_{\Gamma_0, \alpha = \tau, \Gamma_1} \longrightarrow \Delta, \beta : \kappa'} \longrightarrow\!\mathsf{Uvar}$

  $$\begin{aligned}
  (\Gamma_0, \alpha = \tau, \Gamma_1) &= (\Gamma, \beta : \kappa') && \text{Given}\\
  &= (\Gamma_0, \alpha = \tau, \Gamma_1', \beta : \kappa') && \text{Since the final elements must be equal}\\
  \Gamma &= (\Gamma_0, \alpha = \tau, \Gamma_1') && \text{By injectivity of context syntax}\\
  \\
  \Delta &= (\Delta_0, \alpha = \tau', \Delta_1) && \text{By i.h.}\\
  \text{☞}\quad [\Delta_0]\tau &= [\Delta_0]\tau' && ''\\
  \text{☞}\quad \Gamma_0 &\longrightarrow \Delta_0 && ''\\
  \text{☞}\quad (\Delta, \beta : \kappa') &= (\Delta_0, \alpha = \tau', \Delta_1, \beta : \kappa') && \text{By congruence of equality}
  \end{aligned}$$

- **Case** $\dfrac{\Gamma \longrightarrow \Delta \qquad [\Delta]A = [\Delta]A'}{\underbrace{\Gamma, x : A}_{\Gamma_0, \alpha = \tau, \Gamma_1} \longrightarrow \Delta, x : A'} \longrightarrow\!\mathsf{Var}$

  Similar to the $\longrightarrow$Uvar case.

- **Case** $\dfrac{\Gamma \longrightarrow \Delta}{\Gamma, \blacktriangleright_{\hat{\alpha}} \longrightarrow \Delta, \blacktriangleright_{\hat{\alpha}}} \longrightarrow\!\mathsf{Marker}$

  Similar to the $\longrightarrow$Uvar case.

- **Case** $\dfrac{\Gamma \longrightarrow \Delta}{\Gamma, \hat{\alpha} : \kappa' \longrightarrow \Delta, \hat{\alpha} : \kappa'} \longrightarrow\!\mathsf{Unsolved}$

  Similar to the $\longrightarrow$Uvar case.

- **Case** $\dfrac{\Gamma \longrightarrow \Delta \qquad [\Delta]t = [\Delta]t'}{\underbrace{\Gamma, \hat{\alpha} : \kappa' = t}_{\Gamma_0, \alpha = \tau, \Gamma_1} \longrightarrow \Delta, \hat{\alpha} : \kappa' = t'} \longrightarrow\!\mathsf{Solved}$

  Similar to the $\longrightarrow$Uvar case.

- **Case** $\dfrac{\Gamma \longrightarrow \Delta}{\underbrace{\Gamma, \hat{\beta} : \kappa'}_{\Gamma_0, \alpha = \tau, \Gamma_1} \longrightarrow \Delta, \hat{\beta} : \kappa' = t} \longrightarrow\!\mathsf{Solve}$

  Similar to the $\longrightarrow$Uvar case.

- **Case** $\dfrac{\Gamma \longrightarrow \Delta \qquad [\Delta]t = [\Delta]t'}{\underbrace{\Gamma, \beta = t}_{\Gamma_0, \alpha = \tau, \Gamma_1} \longrightarrow \Delta, \beta = t'} \longrightarrow\!\mathsf{Eqn}$

  There are two cases:

  - **Case** $\alpha = \beta$:

    $$\begin{aligned}
    \tau = t \text{ and } \Gamma_1 = \cdot \text{ and } \Gamma_0 = \Gamma && \text{By injectivity of syntax}\\
    \text{☞}\quad \Gamma_0 \longrightarrow \Delta_0 && \text{Subderivation } (\Gamma_0 = \Gamma \text{ and let } \Delta_0 = \Delta)\\
    \text{☞}\quad (\Delta, \alpha = t') = (\Delta_0, \alpha = t', \Delta_1) && \text{where } \Delta_1 = \cdot\\
    \text{☞}\quad [\Delta_0]t = [\Delta_0]t' && \text{By premise } [\Delta]t = [\Delta]t'
    \end{aligned}$$

  - **Case** $\alpha \neq \beta$:

    $$\begin{aligned}
    (\Gamma_0, \alpha = \tau, \Gamma_1) &= (\Gamma, \beta = t) && \text{Given}\\
    &= (\Gamma_0, \alpha = \tau, \Gamma_1', \beta = t) && \text{Since the final elements must be equal}\\
    \Gamma &= (\Gamma_0, \alpha = \tau, \Gamma_1') && \text{By injectivity of context syntax}\\
    \\
    \Delta &= (\Delta_0, \alpha = \tau', \Delta_1) && \text{By i.h.}\\
    \text{☞}\quad [\Delta_0]\tau &= [\Delta_0]\tau' && ''\\
    \text{☞}\quad \Gamma_0 &\longrightarrow \Delta_0 && ''\\
    \text{☞}\quad (\Delta, \beta = t') &= (\Delta_0, \alpha = \tau', \Delta_1, \beta = t') && \text{By congruence of equality}
    \end{aligned}$$

- **Case** $\dfrac{\Gamma \longrightarrow \Delta}{\underbrace{\Gamma}_{\Gamma_0,\alpha=\tau,\Gamma_1} \longrightarrow \Delta, \hat{\alpha} : \kappa'} \longrightarrow\text{Add}$

  $\Delta = (\Delta_0, \alpha = \tau', \Delta_1)$ By i.h.
  ☞ $[\Delta_0]\tau = [\Delta_0]\tau'$ ″
  ☞ $\Gamma_0 \longrightarrow \Delta_0$ ″
  ☞ $(\Delta, \hat{\alpha} : \kappa') = (\Delta_0, \alpha = \tau', \Delta_1, \hat{\alpha} : \kappa')$ By congruence of equality

- **Case** $\dfrac{\Gamma \longrightarrow \Delta}{\underbrace{\Gamma}_{\Gamma_0,\alpha=\tau,\Gamma_1} \longrightarrow \Delta, \hat{\alpha} : \kappa' = t} \longrightarrow\text{AddSolved}$

  $\Delta = (\Delta_0, \alpha = \tau', \Delta_1)$ By i.h.
  ☞ $[\Delta_0]\tau = [\Delta_0]\tau'$ ″
  ☞ $\Gamma_0 \longrightarrow \Delta_0$ ″
  ☞ $(\Delta, \hat{\alpha} : \kappa' = t) = (\Delta_0, \alpha = \tau', \Delta_1, \hat{\alpha} : \kappa' = t)$ By congruence of equality

(iv) We have $\Gamma_0, \hat{\alpha} : \kappa = \tau, \Gamma_1 \longrightarrow \Delta$.

- **Case** $\dfrac{\Gamma \longrightarrow \Delta}{\underbrace{\Gamma, \beta : \kappa'}_{\Gamma_0,\hat{\alpha}:\kappa=\tau,\Gamma_1} \longrightarrow \Delta, \beta : \kappa'} \longrightarrow\text{Uvar}$

  $(\Gamma_0, \hat{\alpha} : \kappa = \tau, \Gamma_1) = (\Gamma, \beta : \kappa')$ Given
  $= (\Gamma_0, \hat{\alpha} : \kappa = \tau, \Gamma_1', \beta : \kappa')$ Since the final elements must be equal
  $\Gamma = (\Gamma_0, \hat{\alpha} : \kappa = \tau, \Gamma_1')$ By injectivity of context syntax

  $\Delta = (\Delta_0, \hat{\alpha} : \kappa = \tau', \Delta_1)$ By i.h.
  ☞ $[\Delta_0]\tau = [\Delta_0]\tau'$ ″
  ☞ $\Gamma_0 \longrightarrow \Delta_0$ ″
  ☞ $(\Delta, \beta : \kappa') = (\Delta_0, \hat{\alpha} : \kappa = \tau', \Delta_1, \beta : \kappa')$ By congruence of equality

- **Case** $\dfrac{\Gamma \longrightarrow \Delta \qquad [\Delta]A = [\Delta]A'}{\underbrace{\Gamma, x : A}_{\Gamma_0,\hat{\alpha}:\kappa=\tau,\Gamma_1} \longrightarrow \Delta, x : A'} \longrightarrow\text{Var}$

  Similar to the $\longrightarrow$Uvar case.

- **Case** $\dfrac{\Gamma \longrightarrow \Delta}{\Gamma, \blacktriangleright_{\hat{\beta}} \longrightarrow \Delta, \blacktriangleright_{\hat{\beta}}} \longrightarrow\text{Marker}$

  Similar to the $\longrightarrow$Uvar case.

- **Case** $\dfrac{\Gamma \longrightarrow \Delta}{\Gamma, \hat{\beta} : \kappa' \longrightarrow \Delta, \hat{\beta} : \kappa'} \longrightarrow\text{Unsolved}$

  Similar to the $\longrightarrow$Uvar case.

- **Case** $\dfrac{\Gamma \longrightarrow \Delta \qquad [\Delta]t = [\Delta]t'}{\underbrace{\Gamma, \hat{\beta} : \kappa' = t}_{\Gamma_0,\hat{\alpha}:\kappa=\tau,\Gamma_1} \longrightarrow \Delta, \hat{\beta} : \kappa' = t'} \longrightarrow\text{Solved}$

  There are two cases.

  – **Case** $\hat{\alpha} = \hat{\beta}$:

    $\kappa' = \kappa$ and $t = \tau$ and $\Gamma_1 = \cdot$ and $\Gamma = \Gamma_0$ By injectivity of syntax
    ☞ $(\Delta, \hat{\beta} : \kappa' = t') = (\Delta_0, \hat{\beta} : \kappa' = \tau', \Delta_1)$ where $\tau' = t'$ and $\Delta_1 = \cdot$ and $\Delta = \Delta_0$
    ☞ $\Gamma_0 \longrightarrow \Delta_0$ From subderivation $\Gamma \longrightarrow \Delta$
    ☞ $[\Delta_0]\tau = [\Delta_0]\tau'$ From premise $[\Delta]t = [\Delta]t'$ and x

– **Case** $\hat{\alpha} \neq \hat{\beta}$:

$$
\begin{array}{rll}
(\Gamma_0, \hat{\alpha} : \kappa = \tau, \Gamma_1) = (\Gamma, \hat{\beta} : \kappa' = t) & & \text{Given} \\
= (\Gamma_0, \hat{\alpha} : \kappa = \tau, \Gamma_1', \hat{\beta} : \kappa' = t) & & \text{Since the final elements must be equal} \\
\Gamma = (\Gamma_0, \hat{\alpha} : \kappa = \tau, \Gamma_1') & & \text{By injectivity of context syntax} \\
\Delta = (\Delta_0, \hat{\alpha} : \kappa = \tau', \Delta_1) & & \text{By i.h.} \\
\text{☞} \quad [\Delta_0]\tau = [\Delta_0]\tau' & & \text{″} \\
\text{☞} \quad \Gamma_0 \longrightarrow \Delta_0 & & \text{″} \\
\text{☞} \quad (\Delta, \hat{\beta} : \kappa' = t') = (\Delta_0, \hat{\alpha} : \kappa = \tau', \Delta_1, \hat{\beta} : \kappa' = t') & & \text{By congruence of equality}
\end{array}
$$

- **Case**
$$\frac{\Gamma \longrightarrow \Delta \qquad [\Delta]t = [\Delta]t'}{\underbrace{\Gamma, \beta = t}_{\Gamma_0, \hat{\alpha}:\kappa=\tau, \Gamma_1} \longrightarrow \Delta, \beta = t'} \longrightarrow \mathsf{Eqn}$$

$$
\begin{array}{rll}
(\Gamma_0, \hat{\alpha} : \kappa = \tau, \Gamma_1) = (\Gamma, \beta = t) & & \text{Given} \\
= (\Gamma_0, \hat{\alpha} : \kappa = \tau, \Gamma_1', \beta = t) & & \text{Since the final elements must be equal} \\
\Gamma = (\Gamma_0, \hat{\alpha} : \kappa = \tau, \Gamma_1') & & \text{By injectivity of context syntax} \\
\Delta = (\Delta_0, \hat{\alpha} : \kappa = \tau', \Delta_1) & & \text{By i.h.} \\
\text{☞} \quad [\Delta_0]\tau = [\Delta_0]\tau' & & \text{″} \\
\text{☞} \quad \Gamma_0 \longrightarrow \Delta_0 & & \text{″} \\
\text{☞} \quad (\Delta, \beta = t') = (\Delta_0, \hat{\alpha} : \kappa = \tau', \Delta_1, \beta = t') & & \text{By congruence of equality}
\end{array}
$$

- **Case**
$$\frac{\Gamma \longrightarrow \Delta}{\underbrace{\Gamma}_{\Gamma_0, \hat{\alpha}:\kappa=\tau, \Gamma_1} \longrightarrow \Delta, \hat{\beta} : \kappa'} \longrightarrow \mathsf{Add}$$

$$
\begin{array}{rll}
\Delta = (\Delta_0, \hat{\alpha} : \kappa = \tau', \Delta_1) & & \text{By i.h.} \\
\text{☞} \quad [\Delta_0]\tau = [\Delta_0]\tau' & & \text{″} \\
\text{☞} \quad \Gamma_0 \longrightarrow \Delta_0 & & \text{″} \\
\text{☞} \quad (\Delta, \hat{\beta} : \kappa') = (\Delta_0, \hat{\alpha} : \kappa = \tau', \Delta_1, \hat{\beta} : \kappa') & & \text{By congruence of equality}
\end{array}
$$

- **Case**
$$\frac{\Gamma \longrightarrow \Delta}{\underbrace{\Gamma}_{\Gamma_0, \hat{\alpha}:\kappa=\tau, \Gamma_1} \longrightarrow \Delta, \hat{\beta} : \kappa' = t} \longrightarrow \mathsf{AddSolved}$$

$$
\begin{array}{rll}
\Delta = (\Delta_0, \hat{\alpha} : \kappa = \tau', \Delta_1) & & \text{By i.h.} \\
\text{☞} \quad [\Delta_0]\tau = [\Delta_0]\tau' & & \text{″} \\
\text{☞} \quad \Gamma_0 \longrightarrow \Delta_0 & & \text{″} \\
\text{☞} \quad (\Delta, \hat{\beta} : \kappa' = t) = (\Delta_0, \hat{\alpha} : \kappa = \tau', \Delta_1, \hat{\beta} : \kappa' = t) & & \text{By congruence of equality}
\end{array}
$$

- **Case**
$$\frac{\Gamma \longrightarrow \Delta}{\underbrace{\Gamma, \hat{\beta} : \kappa'}_{\Gamma_0, \hat{\alpha}:\kappa=\tau, \Gamma_1} \longrightarrow \Delta, \hat{\beta} : \kappa' = t} \longrightarrow \mathsf{Solve}$$

$$
\begin{array}{rll}
(\Gamma, \hat{\beta} : \kappa') = (\Gamma_0, \hat{\alpha} : \kappa = \tau, \Gamma_1) & & \text{Given} \\
= (\Gamma_0, \hat{\alpha} : \kappa = \tau, \Gamma_1', \hat{\beta} : \kappa') & & \text{Since the last elements must be equal} \\
\Gamma = (\Gamma_0, \hat{\alpha} : \kappa = \tau, \Gamma_1') & & \text{By injectivity of syntax} \\
\Gamma \longrightarrow \Delta & & \text{Subderivation} \\
\Gamma_0, \hat{\alpha} : \kappa = \tau, \Gamma_1' \longrightarrow \Delta & & \text{By equality} \\
\Delta = (\Delta_0, \hat{\alpha} : \kappa = \tau', \Delta_1) & & \text{By i.h.} \\
\text{☞} \quad [\Delta_0]\tau = [\Delta_0]\tau' & & \text{″} \\
\text{☞} \quad \Gamma_0 \longrightarrow \Delta_0 & & \text{″} \\
\text{☞} \quad (\Delta, \hat{\beta} : \kappa') = (\Delta_0, \hat{\alpha} : \kappa = \tau', \Delta_1, \hat{\beta} : \kappa') & & \text{By congruence of equality}
\end{array}
$$

(v) We have $\Gamma_0, x : A, \Gamma_1 \longrightarrow \Delta$. This proof is similar to the proof of part (i), except for the domain condition, which we handle similarly to part (ii).

(vi) We have $\Gamma_0, \hat{\alpha} : \kappa, \Gamma_1 \longrightarrow \Delta$.

- **Case** $$\frac{\Gamma \longrightarrow \Delta}{\underbrace{\Gamma, \beta : \kappa'}_{\Gamma_0, \hat{\alpha}:\kappa, \Gamma_1} \longrightarrow \Delta, \beta : \kappa'} \longrightarrow\text{Uvar}$$

  $$\begin{aligned} (\Gamma_0, \hat{\alpha} : \kappa, \Gamma_1) &= (\Gamma, \beta : \kappa') && \text{Given} \\ &= (\Gamma_0, \hat{\alpha} : \kappa, \Gamma_1', \beta : \kappa') && \text{Since the final elements must be equal} \\ \Gamma &= (\Gamma_0, \hat{\alpha} : \kappa, \Gamma_1') && \text{By injectivity of context syntax} \end{aligned}$$

  By induction, there are two possibilities:

  - $\hat{\alpha}$ is not solved:

    $$\begin{aligned} \Delta &= (\Delta_0, \hat{\alpha} : \kappa, \Delta_1) && \text{By i.h.} \\ \text{☞}\quad \Gamma_0 &\longrightarrow \Delta_0 && '' \\ \text{☞}\quad (\Delta, \beta : \kappa') &= (\Delta_0, \hat{\alpha} : \kappa, \Delta_1, \beta : \kappa') && \text{By congruence of equality} \end{aligned}$$

  - $\hat{\alpha}$ is solved:

    $$\begin{aligned} \Delta &= (\Delta_0, \hat{\alpha} : \kappa = \tau', \Delta_1) && \text{By i.h.} \\ \text{☞}\quad \Gamma_0 &\longrightarrow \Delta_0 && '' \\ \text{☞}\quad (\Delta, \beta : \kappa') &= (\Delta_0, \hat{\alpha} : \kappa = \tau', \Delta_1, \beta : \kappa') && \text{By congruence of equality} \end{aligned}$$

- **Case** $$\frac{\Gamma \longrightarrow \Delta \qquad [\Delta]A = [\Delta]A'}{\underbrace{\Gamma, x : A}_{\Gamma_0, \hat{\alpha}:\kappa, \Gamma_1} \longrightarrow \Delta, x : A'} \longrightarrow\text{Var}$$

  Similar to the $\longrightarrow$Uvar case.

- **Case** $$\frac{\Gamma \longrightarrow \Delta}{\Gamma, \blacktriangleright_{\hat{\beta}} \longrightarrow \Delta, \blacktriangleright_{\hat{\beta}}} \longrightarrow\text{Marker}$$

  Similar to the $\longrightarrow$Uvar case.

- **Case** $$\frac{\Gamma \longrightarrow \Delta \qquad [\Delta]t = [\Delta]t'}{\Gamma, \beta = t \longrightarrow \Delta, \beta = t'} \longrightarrow\text{Eqn}$$

  Similar to the $\longrightarrow$Uvar case.

- **Case** $$\frac{\Gamma \longrightarrow \Delta \qquad [\Delta]t = [\Delta]t'}{\underbrace{\Gamma, \hat{\beta} : \kappa' = t}_{\Gamma_0, \hat{\alpha}:\kappa, \Gamma_1} \longrightarrow \Delta, \hat{\beta} : \kappa' = t'} \longrightarrow\text{Solved}$$

  Similar to the $\longrightarrow$Uvar case.

- **Case** $$\frac{\Gamma \longrightarrow \Delta}{\underbrace{\Gamma, \hat{\beta} : \kappa'}_{\Gamma_0, \hat{\alpha}:\kappa, \Gamma_1} \longrightarrow \Delta, \hat{\beta} : \kappa'} \longrightarrow\text{Unsolved}$$

  - **Case** $\hat{\alpha} \neq \hat{\beta}$:

    $$\begin{aligned} (\Gamma_0, \hat{\alpha} : \kappa, \Gamma_1) &= (\Gamma, \hat{\beta} : \kappa') && \text{Given} \\ &= (\Gamma_0, \hat{\alpha} : \kappa, \Gamma_1', \hat{\beta} : \kappa') && \text{Since the final elements must be equal} \\ \Gamma &= (\Gamma_0, \hat{\alpha} : \kappa, \Gamma_1') && \text{By injectivity of context syntax} \end{aligned}$$

    By induction, there are two possibilities:

    * $\hat{\alpha}$ is not solved:

      $$\begin{aligned} \Delta &= (\Delta_0, \hat{\alpha} : \kappa, \Delta_1) && \text{By i.h.} \\ \text{☞}\quad \Gamma_0 &\longrightarrow \Delta_0 && '' \\ \text{☞}\quad (\Delta, \hat{\beta} : \kappa') &= (\Delta_0, \hat{\alpha} : \kappa, \Delta_1, \hat{\beta} : \kappa') && \text{By congruence of equality} \end{aligned}$$

    * $\hat{\alpha}$ is solved:

      $$\begin{aligned} \Delta &= (\Delta_0, \hat{\alpha} : \kappa = \tau', \Delta_1) && \text{By i.h.} \\ \text{☞}\quad \Gamma_0 &\longrightarrow \Delta_0 && '' \\ \text{☞}\quad (\Delta, \hat{\beta} : \kappa') &= (\Delta_0, \hat{\alpha} : \kappa = \tau', \Delta_1, \hat{\beta} : \kappa') && \text{By congruence of equality} \end{aligned}$$

– **Case** $\hat{\alpha} = \hat{\beta}$:

$$\kappa' = \kappa \text{ and } \Gamma_0 = \Gamma \text{ and } \Gamma_1 = \cdot \quad \text{By injectivity of syntax}$$
☞ $(\Delta, \hat{\beta} : \kappa') = (\Delta_0, \hat{\alpha} : \kappa, \Delta_1)$ where $\Delta_0 = \Delta$ and $\Delta_1 = \cdot$
☞ $\Gamma_0 \longrightarrow \Delta_0$ From premise $\Gamma \longrightarrow \Delta$

- **Case**
$$\frac{\Gamma \longrightarrow \Delta}{\underbrace{\Gamma}_{\Gamma_0, \hat{\alpha}:\kappa, \Gamma_1} \longrightarrow \Delta, \hat{\beta} : \kappa'} \longrightarrow\mathsf{Add}$$

By induction, there are two possibilities:

– $\hat{\alpha}$ is not solved:

$$\Delta = (\Delta_0, \hat{\alpha} : \kappa, \Delta_1) \quad \text{By i.h.}$$
☞ $\Gamma_0 \longrightarrow \Delta_0$ ″
☞ $(\Delta, \hat{\beta} : \kappa') = (\Delta_0, \hat{\alpha} : \kappa, \Delta_1, \hat{\beta} : \kappa')$ By congruence of equality

– $\hat{\alpha}$ is solved:

$$\Delta = (\Delta_0, \hat{\alpha} : \kappa = \tau', \Delta_1) \quad \text{By i.h.}$$
☞ $\Gamma_0 \longrightarrow \Delta_0$ ″
☞ $(\Delta, \hat{\beta} : \kappa') = (\Delta_0, \hat{\alpha} : \kappa = \tau', \Delta_1, \hat{\beta} : \kappa')$ By congruence of equality

- **Case**
$$\frac{\Gamma \longrightarrow \Delta}{\underbrace{\Gamma}_{\Gamma_0, \hat{\alpha}:\kappa, \Gamma_1} \longrightarrow \Delta, \hat{\beta} : \kappa' = t} \longrightarrow\mathsf{AddSolved}$$

By induction, there are two possibilities:

– $\hat{\alpha}$ is not solved:

$$\Delta = (\Delta_0, \hat{\alpha} : \kappa, \Delta_1) \quad \text{By i.h.}$$
☞ $\Gamma_0 \longrightarrow \Delta_0$ ″
☞ $(\Delta, \hat{\beta} : \kappa' = t) = (\Delta_0, \hat{\alpha} : \kappa, \Delta_1, \hat{\beta} : \kappa' = t)$ By congruence of equality

– $\hat{\alpha}$ is solved:

$$\Delta = (\Delta_0, \hat{\alpha} : \kappa = \tau', \Delta_1) \quad \text{By i.h.}$$
☞ $\Gamma_0 \longrightarrow \Delta_0$ ″
☞ $(\Delta, \hat{\beta} : \kappa' = t) = (\Delta_0, \hat{\alpha} : \kappa = \tau', \Delta_1, \hat{\beta} : \kappa' = t)$ By congruence of equality

- **Case**
$$\frac{\Gamma \longrightarrow \Delta}{\underbrace{\Gamma, \hat{\beta} : \kappa'}_{\Gamma_0, \hat{\alpha}:\kappa, \Gamma_1} \longrightarrow \Delta, \hat{\beta} : \kappa' = t} \longrightarrow\mathsf{Solve}$$

– **Case** $\hat{\alpha} \neq \hat{\beta}$:

$$(\Gamma_0, \hat{\alpha} : \kappa, \Gamma_1) = (\Gamma, \hat{\beta} : \kappa') \quad \text{Given}$$
$$= (\Gamma_0, \hat{\alpha} : \kappa, \Gamma_1', \hat{\beta} : \kappa') \quad \text{Since the final elements must be equal}$$
$$\Gamma = (\Gamma_0, \hat{\alpha} : \kappa, \Gamma_1') \quad \text{By injectivity of context syntax}$$

By induction, there are two possibilities:

∗ $\hat{\alpha}$ is not solved:

$$\Delta = (\Delta_0, \hat{\alpha} : \kappa, \Delta_1) \quad \text{By i.h.}$$
☞ $\Gamma_0 \longrightarrow \Delta_0$ ″
☞ $(\Delta, \hat{\beta} : \kappa' = t) = (\Delta_0, \hat{\alpha} : \kappa, \Delta_1, \hat{\beta} : \kappa' = t)$ By congruence of equality

∗ $\hat{\alpha}$ is solved:

$$\Delta = (\Delta_0, \hat{\alpha} : \kappa = \tau', \Delta_1) \quad \text{By i.h.}$$
☞ $\Gamma_0 \longrightarrow \Delta_0$ ″
☞ $(\Delta, \hat{\beta} : \kappa' = t) = (\Delta_0, \hat{\alpha} : \kappa = \tau', \Delta_1, \hat{\beta} : \kappa' = t)$ By congruence of equality

– **Case** $\hat{\alpha} = \hat{\beta}$:

$$\Gamma = \Gamma_0 \text{ and } \kappa = \kappa' \text{ and } \Gamma_1 = \cdot \quad \text{By injectivity of syntax}$$
☞ $(\Delta, \hat{\beta} : \kappa' = t) = (\Delta_0, \hat{\alpha} : \kappa = \tau', \Delta_1)$ where $\Delta_0 = \Delta$ and $\tau' = t$ and $\Delta_1 = \cdot$
☞ $\Gamma_0 \longrightarrow \Delta_0$ From premise $\Gamma \longrightarrow \Delta$ □

Lemma 23 (Deep Evar Introduction)**.** *(i) If* Γ_0, Γ_1 *is well-formed and* $\hat{\alpha}$ *is not declared in* Γ_0, Γ_1 *then* $\Gamma_0, \Gamma_1 \longrightarrow \Gamma_0, \hat{\alpha} : \kappa, \Gamma_1$.

(ii) If $\Gamma_0, \hat{\alpha} : \kappa, \Gamma_1$ *is well-formed and* $\Gamma \vdash t : \kappa$ *then* $\Gamma_0, \hat{\alpha} : \kappa, \Gamma_1 \longrightarrow \Gamma_0, \hat{\alpha} : \kappa = t, \Gamma_1$.

(iii) If Γ_0, Γ_1 *is well-formed and* $\Gamma \vdash t : \kappa$ *then* $\Gamma_0, \Gamma_1 \longrightarrow \Gamma_0, \hat{\alpha} : \kappa = t, \Gamma_1$.

Proof.

(i) Assume that Γ_0, Γ_1 is well-formed. We proceed by induction on Γ_1.

- Case $\Gamma_1 = \cdot$:

	Γ_0 *ctx*	Given
	$\hat{\alpha} \notin \mathsf{dom}(\Gamma_0)$	Given
	$\Gamma_0, \hat{\alpha} : \kappa$ *ctx*	By rule VarCtx
$\Gamma_0 \longrightarrow \Gamma_0$		By Lemma 32 (Extension Reflexivity)
☞ $\Gamma_0 \longrightarrow \Gamma_0, \hat{\alpha} : \kappa$		By rule $\longrightarrow$Add

- Case $\Gamma_1 = \Gamma_1', x : A$:

	$\Gamma_0, \Gamma_1', x : A$ *ctx*	Given
	Γ_0, Γ_1' *ctx*	By inversion
	$x \notin \mathsf{dom}(\Gamma_0, \Gamma_1')$	By inversion (1)
$\Gamma_0, \Gamma_1' \vdash$	A *type*	By inversion
	$\hat{\alpha} \notin \mathsf{dom}(\Gamma_0, \Gamma_1', x : A)$	Given
	$\hat{\alpha} \neq x$	By inversion (2)
	$\Gamma_0, \hat{\alpha} : \kappa, \Gamma_1'$ *ctx*	By i.h.
$\Gamma_0, \Gamma_1' \longrightarrow$	$\Gamma_0, \hat{\alpha} : \kappa, \Gamma_1'$	$''$
$\Gamma_0, \hat{\alpha} : \kappa, \Gamma_1' \vdash$	A *type*	By Lemma 36 (Extension Weakening (Sorts))
	$x \notin \mathsf{dom}(\Gamma_0, \hat{\alpha} : \kappa, \Gamma_1')$	By (1) and (2)
☞ $\Gamma_0, \Gamma_1', x : A \longrightarrow$	$\Gamma_0, \hat{\alpha} : \kappa, \Gamma_1', x : A$	By $\longrightarrow$Var

- Case $\Gamma_1 = \Gamma_1', \beta : \kappa'$:

	$\Gamma_0, \Gamma_1', \beta : \kappa'$ *ctx*	Given
	Γ_0, Γ_1' *ctx*	By inversion
	$\beta \notin \mathsf{dom}(\Gamma_0, \Gamma_1')$	By inversion (1)
	$\hat{\alpha} \notin \mathsf{dom}(\Gamma_0, \Gamma_1', \beta : \kappa')$	Given
	$\hat{\alpha} \neq \beta$	By inversion (2)
	$\Gamma_0, \hat{\alpha} : \kappa, \Gamma_1'$ *ctx*	By i.h.
$\Gamma_0, \Gamma_1' \longrightarrow$	$\Gamma_0, \hat{\alpha} : \kappa, \Gamma_1'$	$''$
	$\beta \notin \mathsf{dom}(\Gamma_0, \hat{\alpha} : \kappa, \Gamma_1')$	By (1) and (2)
☞ $\Gamma_0, \Gamma_1', \beta : \kappa' \longrightarrow$	$\Gamma_0, \hat{\alpha} : \kappa, \Gamma_1', \beta : \kappa'$	By $\longrightarrow$Uvar

- Case $\Gamma_1 = \Gamma_1', \hat{\beta} : \kappa'$:

	$\Gamma_0, \Gamma_1', \hat{\beta} : \kappa'$ *ctx*	Given
	Γ_0, Γ_1' *ctx*	By inversion
	$\hat{\beta} \notin \mathsf{dom}(\Gamma_0, \Gamma_1')$	By inversion (1)
	$\hat{\alpha} \notin \mathsf{dom}(\Gamma_0, \Gamma_1', \hat{\beta} : \kappa')$	Given
	$\hat{\alpha} \neq \hat{\beta}$	By inversion (2)
	$\Gamma_0, \hat{\alpha} : \kappa, \Gamma_1'$ *ctx*	By i.h.
$\Gamma_0, \Gamma_1' \longrightarrow$	$\Gamma_0, \hat{\alpha} : \kappa, \Gamma_1'$	$''$
	$\hat{\beta} \notin \mathsf{dom}(\Gamma_0, \hat{\alpha} : \kappa, \Gamma_1')$	By (1) and (2)
☞ $\Gamma_0, \Gamma_1', \hat{\beta} : \kappa' \longrightarrow$	$\Gamma_0, \hat{\alpha} : \kappa, \Gamma_1', \hat{\beta} : \kappa'$	By $\longrightarrow$Unsolved

- Case $\Gamma_1 = (\Gamma_1', \hat{\beta} : \kappa' = t)$:

$\Gamma_0, \Gamma_1', \hat{\beta} : \kappa' = t$ *ctx* Given
Γ_0, Γ_1' *ctx* By inversion
$\hat{\beta} \notin \mathsf{dom}(\Gamma_0, \Gamma_1')$ By inversion (1)
$\Gamma_0, \Gamma_1' \vdash t : \kappa'$ By inversion
$\hat{\alpha} \notin \mathsf{dom}(\Gamma_0, \Gamma_1', \hat{\beta} : \kappa' = t)$ Given
$\hat{\alpha} \neq \hat{\beta}$ By inversion (2)
$\Gamma_0, \hat{\alpha} : \kappa, \Gamma_1'$ *ctx* By i.h.
$\Gamma_0, \Gamma_1' \longrightarrow \Gamma_0, \hat{\alpha} : \kappa, \Gamma_1'$ ″
$\Gamma_0, \hat{\alpha} : \kappa, \Gamma_1' \vdash t : \kappa'$ By Lemma 36 (Extension Weakening (Sorts))
$\hat{\beta} \notin \mathsf{dom}(\Gamma_0, \hat{\alpha} : \kappa, \Gamma_1')$ By (1) and (2)
☞ $\Gamma_0, \Gamma_1', \hat{\beta} : \kappa' = t \longrightarrow \Gamma_0, \hat{\alpha} : \kappa, \Gamma_1', \hat{\beta} : \kappa' = t$ By $\longrightarrow$Solved

- Case $\Gamma_1 = (\Gamma_1', \beta = t)$:

$\Gamma_0, \Gamma_1', \beta = t$ *ctx* Given
Γ_0, Γ_1' *ctx* By inversion
$\beta \notin \mathsf{dom}(\Gamma_0, \Gamma_1')$ By inversion (1)
$\Gamma_0, \Gamma_1' \vdash t : \mathbb{N}$ By inversion
$\hat{\alpha} \notin \mathsf{dom}(\Gamma_0, \Gamma_1', \beta = t)$ Given
$\hat{\alpha} \neq \beta$ By inversion (2)
$\Gamma_0, \hat{\alpha} : \kappa, \Gamma_1'$ *ctx* By i.h.
$\Gamma_0, \Gamma_1' \longrightarrow \Gamma_0, \hat{\alpha} : \kappa, \Gamma_1'$ ″
$\Gamma_0, \hat{\alpha} : \kappa, \Gamma_1' \vdash t : \mathbb{N}$ By Lemma 36 (Extension Weakening (Sorts))
$\beta \notin \mathsf{dom}(\Gamma_0, \hat{\alpha} : \kappa, \Gamma_1')$ By (1) and (2)
☞ $\Gamma_0, \Gamma_1', \beta = t \longrightarrow \Gamma_0, \hat{\alpha} : \kappa, \Gamma_1', \beta = t$ By $\longrightarrow$Solved

- Case $\Gamma_1 = (\Gamma_1', \blacktriangleright_{\hat{\beta}})$:

$\Gamma_0, \Gamma_1', \blacktriangleright_{\hat{\beta}}$ *ctx* Given
Γ_0, Γ_1' *ctx* By inversion
$\hat{\beta} \notin \mathsf{dom}(\Gamma_0, \Gamma_1')$ By inversion (1)
$\hat{\alpha} \notin \mathsf{dom}(\Gamma_0, \Gamma_1', \blacktriangleright_{\hat{\beta}})$ Given
$\hat{\alpha} \neq \hat{\beta}$ By inversion (2)
$\Gamma_0, \hat{\alpha} : \kappa, \Gamma_1'$ *ctx* By i.h.
$\Gamma_0, \Gamma_1' \longrightarrow \Gamma_0, \hat{\alpha} : \kappa, \Gamma_1'$ ″
$\hat{\beta} \notin \mathsf{dom}(\Gamma_0, \hat{\alpha} : \kappa, \Gamma_1')$ By (1) and (2)
☞ $\Gamma_0, \Gamma_1', \blacktriangleright_{\hat{\beta}} \longrightarrow \Gamma_0, \hat{\alpha} : \kappa, \Gamma_1', \blacktriangleright_{\hat{\beta}}$ By $\longrightarrow$Marker

(ii) Assume $\Gamma_0, \hat{\alpha} : \kappa, \Gamma_1$ *ctx*. We proceed by induction on Γ_1:

- Case $\Gamma_1 = \cdot$:

$\Gamma_0 \vdash t : \kappa$ Given
Γ_0, Γ_1 *ctx* Given
Γ_0 *ctx* Since $\Gamma_1 = \cdot$
$\Gamma_0 \longrightarrow \Gamma_0$ By Lemma 32 (Extension Reflexivity)
$\Gamma_0, \hat{\alpha} : \kappa \longrightarrow \Gamma_0, \hat{\alpha} : \kappa = t$ By rule $\longrightarrow$Solve
☞ $\Gamma_0, \hat{\alpha} : \kappa, \Gamma_1 \longrightarrow \Gamma_0, \hat{\alpha} : \kappa = t, \Gamma_1$ Since $\Gamma_1 = \cdot$

- Case $\Gamma_1 = (\Gamma_1', x : A)$:

$\Gamma_0 \vdash t : \kappa$ Given
$\Gamma_0, \hat{\alpha} : \kappa, \Gamma_1', x : A$ *ctx* Given
$\Gamma_0, \hat{\alpha} : \kappa, \Gamma_1'$ *ctx* By inversion
$\Gamma_0, \hat{\alpha} : \kappa, \Gamma_1' \vdash A$ *type* By inversion
$x \notin \mathsf{dom}(\Gamma_0, \hat{\alpha} : \kappa, \Gamma_1')$ By inversion (1)
$\Gamma_0, \hat{\alpha} : \kappa, \Gamma_1' \longrightarrow \Gamma_0, \hat{\alpha} : \kappa = t, \Gamma_1$ By i.h.
$\Gamma_0, \hat{\alpha} : \kappa = t, \Gamma_1 \vdash A$ *type* By Lemma 36 (Extension Weakening (Sorts))
$x \notin \mathsf{dom}(\Gamma_0, \hat{\alpha} : \kappa = t, \Gamma_1')$ since this is the same domain as (1)
$\Gamma_0, \hat{\alpha} : \kappa, \Gamma_1', x : A \longrightarrow \Gamma_0, \hat{\alpha} : \kappa = t, \Gamma_1, x : A$ By rule $\longrightarrow$Var

- Case $\Gamma_1 = (\Gamma_1', \beta : \kappa')$:

$$\begin{array}{rl}
\Gamma_0 \vdash t : \kappa & \text{Given} \\
\Gamma_0, \hat{\alpha} : \kappa, \Gamma_1', \beta : \kappa' \; \mathit{ctx} & \text{Given} \\
\Gamma_0, \hat{\alpha} : \kappa, \Gamma_1' \; \mathit{ctx} & \text{By inversion} \\
\beta \notin \mathsf{dom}(\Gamma_0, \hat{\alpha} : \kappa, \Gamma_1') & \text{By inversion (1)} \\
\Gamma_0, \hat{\alpha} : \kappa, \Gamma_1' \longrightarrow \Gamma_0, \hat{\alpha} : \kappa = t, \Gamma_1 & \text{By i.h.} \\
\beta \notin \mathsf{dom}(\Gamma_0, \hat{\alpha} : \kappa = t, \Gamma_1') & \text{since this is the same domain as (1)} \\
\Gamma_0, \hat{\alpha} : \kappa, \Gamma_1', \beta : \kappa' \longrightarrow \Gamma_0, \hat{\alpha} : \kappa = t, \Gamma_1, \beta : \kappa' & \text{By rule } \longrightarrow\mathsf{Uvar}
\end{array}$$

- Case $\Gamma_1 = (\Gamma_1', \hat{\beta} : \kappa')$:

$$\begin{array}{rl}
\Gamma_0 \vdash t : \kappa & \text{Given} \\
\Gamma_0, \hat{\alpha} : \kappa, \Gamma_1', \hat{\beta} : \kappa' \; \mathit{ctx} & \text{Given} \\
\Gamma_0, \hat{\alpha} : \kappa, \Gamma_1' \; \mathit{ctx} & \text{By inversion} \\
\hat{\beta} \notin \mathsf{dom}(\Gamma_0, \hat{\alpha} : \kappa, \Gamma_1') & \text{By inversion (1)} \\
\Gamma_0, \hat{\alpha} : \kappa, \Gamma_1' \longrightarrow \Gamma_0, \hat{\alpha} : \kappa = t, \Gamma_1 & \text{By i.h.} \\
\hat{\beta} \notin \mathsf{dom}(\Gamma_0, \hat{\alpha} : \kappa = t, \Gamma_1') & \text{since this is the same domain as (1)} \\
\Gamma_0, \hat{\alpha} : \kappa, \Gamma_1', \hat{\beta} : \kappa' \longrightarrow \Gamma_0, \hat{\alpha} : \kappa = t, \Gamma_1, \hat{\beta} : \kappa' & \text{By rule } \longrightarrow\mathsf{Unsolved}
\end{array}$$

- Case $\Gamma_1 = (\Gamma_1', \hat{\beta} : \kappa' = t')$:

$$\begin{array}{rl}
\Gamma_0 \vdash t' : \kappa & \text{Given} \\
\Gamma_0, \hat{\alpha} : \kappa, \Gamma_1', \hat{\beta} : \kappa' = t' \; \mathit{ctx} & \text{Given} \\
\Gamma_0, \hat{\alpha} : \kappa, \Gamma_1' \; \mathit{ctx} & \text{By inversion} \\
\Gamma_0, \hat{\alpha} : \kappa, \Gamma_1' \vdash t' : \kappa' & \text{By inversion} \\
\hat{\beta} \notin \mathsf{dom}(\Gamma_0, \hat{\alpha} : \kappa, \Gamma_1') & \text{By inversion (1)} \\
\Gamma_0, \hat{\alpha} : \kappa, \Gamma_1' \longrightarrow \Gamma_0, \hat{\alpha} : \kappa = t, \Gamma_1 & \text{By i.h.} \\
\hat{\beta} \notin \mathsf{dom}(\Gamma_0, \hat{\alpha} : \kappa = t, \Gamma_1') & \text{since this is the same domain as (1)}
\end{array}$$

$\Gamma_0, \hat{\alpha} : \kappa = t, \Gamma_1 \vdash t' : \kappa'$ By Lemma 36 (Extension Weakening (Sorts))
$\Gamma_0, \hat{\alpha} : \kappa, \Gamma_1', \hat{\beta} : \kappa' = t' \longrightarrow \Gamma_0, \hat{\alpha} : \kappa = t', \Gamma_1, \hat{\beta} : \kappa' = t'$ By rule $\longrightarrow$Solved

- Case $\Gamma_1 = (\Gamma_1', \beta = t')$:

$$\begin{array}{rl}
\Gamma_0 \vdash t' : \kappa & \text{Given} \\
\Gamma_0, \hat{\alpha} : \kappa, \Gamma_1', \beta = t' \; \mathit{ctx} & \text{Given} \\
\Gamma_0, \hat{\alpha} : \kappa, \Gamma_1' \; \mathit{ctx} & \text{By inversion} \\
\Gamma_0, \hat{\alpha} : \kappa, \Gamma_1' \vdash t' : \mathbb{N} & \text{By inversion} \\
\beta \notin \mathsf{dom}(\Gamma_0, \hat{\alpha} : \kappa, \Gamma_1') & \text{By inversion (1)} \\
\Gamma_0, \hat{\alpha} : \kappa, \Gamma_1' \longrightarrow \Gamma_0, \hat{\alpha} : \kappa = t, \Gamma_1 & \text{By i.h.} \\
\beta \notin \mathsf{dom}(\Gamma_0, \hat{\alpha} : \kappa = t, \Gamma_1') & \text{since this is the same domain as (1)}
\end{array}$$

$\Gamma_0, \hat{\alpha} : \kappa = t, \Gamma_1 \vdash t' : \mathbb{N}$ By Lemma 36 (Extension Weakening (Sorts))
$\Gamma_0, \hat{\alpha} : \kappa, \Gamma_1', \beta = t' \longrightarrow \Gamma_0, \hat{\alpha} : \kappa = t', \Gamma_1, \beta = t'$ By rule $\longrightarrow$Eqn

- Case $\Gamma_1 = (\Gamma_1', \blacktriangleright_{\hat{\beta}})$:

$$\begin{array}{rl}
\Gamma_0 \vdash t : \kappa & \text{Given} \\
\Gamma_0, \hat{\alpha} : \kappa, \Gamma_1', \blacktriangleright_{\hat{\beta}} \; \mathit{ctx} & \text{Given} \\
\Gamma_0, \hat{\alpha} : \kappa, \Gamma_1' \; \mathit{ctx} & \text{By inversion} \\
\hat{\beta} \notin \mathsf{dom}(\Gamma_0, \hat{\alpha} : \kappa, \Gamma_1') & \text{By inversion (1)} \\
\Gamma_0, \hat{\alpha} : \kappa, \Gamma_1' \longrightarrow \Gamma_0, \hat{\alpha} : \kappa = t, \Gamma_1 & \text{By i.h.} \\
\hat{\beta} \notin \mathsf{dom}(\Gamma_0, \hat{\alpha} : \kappa = t, \Gamma_1') & \text{since this is the same domain as (1)} \\
\Gamma_0, \hat{\alpha} : \kappa, \Gamma_1', \blacktriangleright_{\hat{\beta}} \longrightarrow \Gamma_0, \hat{\alpha} : \kappa = t, \Gamma_1, \blacktriangleright_{\hat{\beta}} & \text{By rule } \longrightarrow\mathsf{Unsolved}
\end{array}$$

(iii) Apply parts (i) and (ii) as lemmas, then Lemma 33 (Extension Transitivity). □

Lemma 26 (Parallel Admissibility)**.**
If $\Gamma_L \longrightarrow \Delta_L$ *and* $\Gamma_L, \Gamma_R \longrightarrow \Delta_L, \Delta_R$ *then:*

(i) $\Gamma_L, \hat{\alpha} : \kappa, \Gamma_R \longrightarrow \Delta_L, \hat{\alpha} : \kappa, \Delta_R$

(ii) *If* $\Delta_L \vdash \tau' : \kappa$ *then* $\Gamma_L, \hat{\alpha} : \kappa, \Gamma_R \longrightarrow \Delta_L, \hat{\alpha} : \kappa = \tau', \Delta_R$.

(iii) *If* $\Gamma_L \vdash \tau : \kappa$ *and* $\Delta_L \vdash \tau'$ *type and* $[\Delta_L]\tau = [\Delta_L]\tau'$*, then* $\Gamma_L, \hat{\alpha} : \kappa = \tau, \Gamma_R \longrightarrow \Delta_L, \hat{\alpha} : \kappa = \tau', \Delta_R$.

Proof. By induction on Δ_R. As always, we assume that all contexts mentioned in the statement of the lemma are well-formed. Hence, $\hat{\alpha} \notin \mathsf{dom}(\Gamma_L) \cup \mathsf{dom}(\Gamma_R) \cup \mathsf{dom}(\Delta_L) \cup \mathsf{dom}(\Delta_R)$.

(i) We proceed by cases of Δ_R. Observe that in all the extension rules, the right-hand context gets smaller, so as we enter subderivations of $\Gamma_L, \Gamma_R \longrightarrow \Delta_L, \Delta_R$, the context Δ_R becomes smaller.

The only tricky part of the proof is that to apply the i.h., we need $\Gamma_L \longrightarrow \Delta_L$. So we need to make sure that as we drop items from the right of Γ_R and Δ_R, we don't go too far and start decomposing Γ_L or Δ_L! It's easy to avoid decomposing Δ_L: when $\Delta_R = \cdot$, we don't need to apply the i.h. anyway. To avoid decomposing Γ_L, we need to reason by contradiction, using Lemma 19 (Declaration Preservation).

- **Case** $\Delta_R = \cdot$:
  We have $\Gamma_L \longrightarrow \Delta_L$. Applying $\longrightarrow$Unsolved to that derivation gives the result.
- **Case** $\Delta_R = (\Delta_R', \hat{\beta})$: We have $\hat{\beta} \neq \hat{\alpha}$ by the well-formedness assumption.
  The concluding rule of $\Gamma_L, \Gamma_R \longrightarrow \Delta_L, \Delta_R', \hat{\beta}$ must have been $\longrightarrow$Unsolved or $\longrightarrow$Add. In both cases, the result follows by i.h. and applying $\longrightarrow$Unsolved or $\longrightarrow$Add.
  Note: In $\longrightarrow$Add, the left-hand context doesn't change, so we clearly maintain $\Gamma_L \longrightarrow \Delta_L$. In $\longrightarrow$Unsolved, we can correctly apply the i.h. because $\Gamma_R \neq \cdot$. Suppose, for a contradiction, that $\Gamma_R = \cdot$. Then $\Gamma_L = (\Gamma_L', \hat{\beta})$. It was given that $\Gamma_L \longrightarrow \Delta_L$, that is, $\Gamma_L', \hat{\beta} \longrightarrow \Delta_L$. By Lemma 19 (Declaration Preservation), Δ_L has a declaration of $\hat{\beta}$. But then $\Delta = (\Delta_L, \Delta_R', \hat{\beta})$ is not well-formed: contradiction. Therefore $\Gamma_R \neq \cdot$.
- **Case** $\Delta_R = (\Delta_R', \hat{\beta} : \kappa = t)$: We have $\hat{\beta} \neq \hat{\alpha}$ by the well-formedness assumption.
  The concluding rule must have been $\longrightarrow$Solved, $\longrightarrow$Solve or $\longrightarrow$AddSolved. In each case, apply the i.h. and then the corresponding rule. (In $\longrightarrow$Solved and $\longrightarrow$Solve, use Lemma 19 (Declaration Preservation) to show $\Gamma_R \neq \cdot$.)
- **Case** $\Delta_R = (\Delta_R', \alpha)$: The concluding rule must have been $\longrightarrow$Uvar. The result follows by i.h. and applying $\longrightarrow$Uvar.
- **Case** $\Delta_R = (\Delta_R', \alpha = \tau)$: The concluding rule must have been $\longrightarrow$Eqn. The result follows by i.h. and applying $\longrightarrow$Eqn.
- **Case** $\Delta_R = (\Delta_R', \blacktriangleright_{\hat{\beta}})$: Similar to the previous case, with rule $\longrightarrow$Marker.
- **Case** $\Delta_R = (\Delta_R', x : A)$: Similar to the previous case, with rule $\longrightarrow$Var.

(ii) Similar to part (i), except that when $\Delta_R = \cdot$, apply rule $\longrightarrow$Solve.

(iii) Similar to part (i), except that when $\Delta_R = \cdot$, apply rule $\longrightarrow$Solved, using the given equality to satisfy the second premise. □

Lemma 27 (Parallel Extension Solution)**.**
If $\Gamma_L, \hat{\alpha} : \kappa, \Gamma_R \longrightarrow \Delta_L, \hat{\alpha} : \kappa = \tau', \Delta_R$ *and* $\Gamma_L \vdash \tau : \kappa$ *and* $[\Delta_L]\tau = [\Delta_L]\tau'$
then $\Gamma_L, \hat{\alpha} : \kappa = \tau, \Gamma_R \longrightarrow \Delta_L, \hat{\alpha} : \kappa = \tau', \Delta_R$.

Proof. By induction on Δ_R.

In the case where $\Delta_R = \cdot$, we know that rule $\longrightarrow$Solve must have concluded the derivation (we can use Lemma 19 (Declaration Preservation) to get a contradiction that rules out $\longrightarrow$AddSolved); then we have a subderivation $\Gamma_L \longrightarrow \Delta_L$, to which we can apply $\longrightarrow$Solved. □

Lemma 28 (Parallel Variable Update)**.**
If $\Gamma_L, \hat{\alpha} : \kappa, \Gamma_R \longrightarrow \Delta_L, \hat{\alpha} : \kappa = \tau_0, \Delta_R$ *and* $\Gamma_L \vdash \tau_1 : \kappa$ *and* $\Delta_L \vdash \tau_2 : \kappa$ *and* $[\Delta_L]\tau_0 = [\Delta_L]\tau_1 = [\Delta_L]\tau_2$
then $\Gamma_L, \hat{\alpha} : \kappa = \tau_1, \Gamma_R \longrightarrow \Delta_L, \hat{\alpha} : \kappa = \tau_2, \Delta_R$.

Proof. By induction on Δ_R. Similar to the proof of Lemma 27 (Parallel Extension Solution), but applying $\longrightarrow$Solved at the end. □

Lemma 29 (Substitution Monotonicity)**.**

(i) *If* $\Gamma \longrightarrow \Delta$ *and* $\Gamma \vdash\ t : \kappa$ *then* $[\Delta][\Gamma]t = [\Delta]t$.

(ii) *If* $\Gamma \longrightarrow \Delta$ *and* $\Gamma \vdash\ P$ *prop then* $[\Delta][\Gamma]P = [\Delta]P$.

(iii) *If* $\Gamma \longrightarrow \Delta$ *and* $\Gamma \vdash\ A$ *type then* $[\Delta][\Gamma]A = [\Delta]A$.

Proof. We prove each part in turn; part (i) does not depend on parts (ii) or (iii), so we can use part (i) as a lemma in the proofs of parts (ii) and (iii).

- **Proof of Part (i):** By lexicographic induction on the derivation of $\mathcal{D} :: \Gamma \longrightarrow \Delta$ and $\Gamma \vdash\ t : \kappa$. We proceed by cases on the derivation of $\Gamma \vdash\ t : \kappa$.

  - **Case** $\dfrac{\hat{\alpha} : \kappa \in \Gamma}{\Gamma \vdash\ \hat{\alpha} : \kappa}\ \mathsf{VarSort}$

    $$\begin{array}{rll} [\Gamma]\hat{\alpha} = \hat{\alpha} & & \text{Since } \hat{\alpha} \text{ is not solved in } \Gamma \\ [\Delta]\hat{\alpha} = [\Delta]\hat{\alpha} & & \text{Reflexivity} \\ \quad = [\Delta][\Gamma]\hat{\alpha} & & \text{By above equality} \end{array}$$

  - **Case** $\dfrac{(\alpha : \kappa) \in \Gamma}{\Gamma \vdash\ \alpha : \kappa}\ \mathsf{VarSort}$

    Consider whether or not there is a binding of the form $(\alpha = \tau) \in \Gamma$.

    * **Case** $(\alpha = \tau) \in \Gamma$:

      $$\begin{array}{rll} & \Delta = (\Delta_0, \alpha = \tau', \Delta_1) & \text{By Lemma 22 (Extension Inversion) (i)} \\ \mathcal{D}' :: & \Gamma_0 \longrightarrow \Delta_0 & '' \\ & \mathcal{D}' < \mathcal{D} & '' \\ (1) & [\Delta_0]\tau' = [\Delta_0]\tau & '' \\ (2) & [\Delta_0][\Gamma_0]\tau = [\Delta_0]\tau & \text{By i.h.} \\ & [\Delta][\Gamma]\alpha = [\Delta_0, \alpha = \tau', \Delta_1][\Gamma_0, \alpha = \tau, \Gamma_1]\alpha & \text{By definition} \\ & \quad = [\Delta_0, \alpha = \tau', \Delta_1][\Gamma_0, \alpha = \tau]\alpha & \text{Since } \alpha \notin \mathsf{dom}(\Gamma_1) \\ & \quad = [\Delta_0, \alpha = \tau', \Delta_1][\Gamma_0]\tau & \text{By definition of substitution} \\ & \quad = [\Delta_0][\Gamma_0]\tau & \text{Since } \mathrm{FV}([\Gamma_0]\tau) \cap \mathsf{dom}(\Delta_1) = \emptyset \\ & \quad = [\Delta_0]\tau' & \text{By (2) and (1)} \\ & \quad = [\Delta_0, \alpha = \tau']\alpha & \text{By definition of substitution} \\ & \quad = [\Delta_0, \alpha = \tau', \Delta_1]\alpha & \text{Since } \mathrm{FV}([\Delta_0]\tau) \cap \mathsf{dom}(\Delta_1) = \emptyset \\ & \quad = [\Delta]\alpha & \text{By definition of } \Delta \end{array}$$

    * **Case** $(\alpha = \tau) \notin \Gamma$:

      $$\begin{array}{rll} [\Gamma]\alpha = \alpha & & \text{By definition of substitution} \\ [\Delta][\Gamma]\alpha = [\Delta]\alpha & & \text{Apply } [\Delta] \text{ to both sides} \end{array}$$

  - **Case**

    $$\frac{}{\Gamma_0, \hat{\alpha} : \kappa = \tau, \Gamma_1 \vdash\ \hat{\alpha} : \kappa}\ \mathsf{SolvedVarSort}$$

    Similar to the VarSort case.

  - **Case**

    $$\frac{}{\Gamma \vdash\ 1 : \star}\ \mathsf{UnitSort}$$

    $[\Delta]1 = 1 = [\Delta][\Gamma]1$ Since $\mathrm{FV}(1) = \emptyset$

  - **Case** $\dfrac{\Gamma \vdash\ \tau_1 : \star \qquad \Gamma \vdash\ \tau_2 : \star}{\Gamma \vdash\ \tau_1 \oplus \tau_2 : \star}\ \mathsf{BinSort}$

    $$\begin{array}{rll} [\Delta][\Gamma]\tau_1 = [\Delta]\tau_1 & & \text{By i.h.} \\ [\Delta][\Gamma]\tau_2 = [\Delta]\tau_2 & & \text{By i.h.} \\ [\Delta][\Gamma]\tau_1 \oplus [\Delta][\Gamma]\tau_2 = [\Delta]\tau_1 \oplus [\Delta]\tau_2 & & \text{By congruence of equality} \\ [\Delta][\Gamma](\tau_1 \oplus \tau_2) = [\Delta](\tau_1 \oplus \tau_2) & & \text{Definition of substitution} \end{array}$$

– **Case**
$$\frac{}{\Gamma \vdash \mathsf{zero} : \mathbb{N}}\ \mathsf{ZeroSort}$$
$[\Delta]\mathsf{zero} = \mathsf{zero} = [\Delta][\Gamma]\mathsf{zero}$ Since $\mathrm{FV}(\mathsf{zero}) = \emptyset$

– **Case**
$$\frac{\Gamma \vdash t : \mathbb{N}}{\Gamma \vdash \mathsf{succ}(t) : \mathbb{N}}\ \mathsf{SuccSort}$$
$[\Delta][\Gamma]t = [\Delta]t$ By i.h.
$\mathsf{succ}([\Delta][\Gamma]t) = \mathsf{succ}([\Delta]t)$ By congruence of equality
$[\Delta][\Gamma]\mathsf{succ}(t) = [\Delta]\mathsf{succ}(t)$ By definition of substitution

- **Proof of Part (ii):** We have a derivation of $\Gamma \vdash P\ \mathit{prop}$, and will use the previous part as a lemma.

– **Case**
$$\frac{\Gamma \vdash t : \mathbb{N} \qquad \Gamma \vdash t' : \mathbb{N}}{\Gamma \vdash t = t'\ \mathit{prop}}\ \mathsf{EqProp}$$
$[\Delta][\Gamma]t = [\Delta]t$ By part (i)
$[\Delta][\Gamma]t' = [\Delta]t'$ By part (i)
$([\Delta][\Gamma]t = [\Delta][\Gamma]t') = ([\Delta]t = [\Delta]t')$ By congruence of equality
$[\Delta][\Gamma](t = t') = [\Delta](t = t')$ Definition of substitution

- **Proof of Part (iii):** By induction on the derivation of $\Gamma \vdash A\ \mathit{type}$, using the previous parts as lemmas.

– **Case**
$$\frac{(u : \star) \in \Gamma}{\Gamma \vdash u\ \mathit{type}}\ \mathsf{VarWF}$$
$\Gamma \vdash u : \star$ By rule VarSort
$[\Delta][\Gamma]u = [\Delta]u$ By part (i)

– **Case**
$$\frac{(\hat{\alpha} : \star = \tau) \in \Gamma}{\Gamma \vdash \hat{\alpha}\ \mathit{type}}\ \mathsf{SolvedVarWF}$$
$\Gamma \vdash \hat{\alpha} : \star$ By rule SolvedVarSort
$[\Delta][\Gamma]\hat{\alpha} = [\Delta]\hat{\alpha}$ By part (i)

– **Case**
$$\frac{}{\Gamma \vdash 1\ \mathit{type}}\ \mathsf{UnitWF}$$
$\Gamma \vdash 1 : \star$ By rule UnitSort
$[\Delta][\Gamma]1 = [\Delta]1$ By part (i)

– **Case**
$$\frac{\Gamma \vdash A_1\ \mathit{type} \qquad \Gamma \vdash A_2\ \mathit{type}}{\Gamma \vdash A_1 \oplus A_2\ \mathit{type}}\ \mathsf{BinWF}$$
$[\Delta][\Gamma]A_1 = [\Delta]A_1$ By i.h.
$[\Delta][\Gamma]A_2 = [\Delta]A_2$ By i.h.
$[\Delta][\Gamma]A_1 \oplus [\Delta][\Gamma]A_2 = [\Delta]A_1 \oplus [\Delta]A_2$ By congruence of equality
$[\Delta][\Gamma](A_1 \oplus A_2) = [\Delta](A_1 \oplus A_2)$ Definition of substitution

– **Case** VecWF: Similar to the BinWF case.

– **Case**
$$\frac{\Gamma, \alpha : \kappa \vdash A_0\ \mathit{type}}{\Gamma \vdash \forall \alpha : \kappa.\ A_0\ \mathit{type}}\ \mathsf{ForallWF}$$
$\Gamma \longrightarrow \Delta$ Given
$\Gamma, \alpha : \kappa \longrightarrow \Delta, \alpha : \kappa$ By rule $\longrightarrow$Uvar
$[\Delta, \alpha : \kappa][\Gamma, \alpha : \kappa]A_0 = [\Delta, \alpha : \kappa]A_0$ By i.h.
$[\Delta][\Gamma]A_0 = [\Delta]A_0$ By definition of substitution
$\forall \alpha : \kappa.\ [\Delta][\Gamma]A_0 = \forall \alpha : \kappa.\ [\Delta]A_0$ By congruence of equality
$[\Delta][\Gamma](\forall \alpha : \kappa.\ A_0) = [\Delta](\forall \alpha : \kappa.\ A_0)$ By definition of substitution

– **Case** ExistsWF: Similar to the ForallWF case.

– **Case** $\dfrac{\Gamma \vdash P\ \mathit{prop} \qquad \Gamma \vdash A_0\ \mathit{type}}{\Gamma \vdash P \supset A_0\ \mathit{type}}\ \mathsf{ImpliesWF}$

$$\begin{aligned} [\Delta][\Gamma]P &= [\Delta]P && \text{By part (ii)}\\ [\Delta][\Gamma]A_0 &= [\Delta]A_0 && \text{By i.h.}\\ [\Delta][\Gamma]P \supset [\Delta][\Gamma]A_0 &= [\Delta]P \supset [\Delta]A_0 && \text{By congruence of equality}\\ [\Delta][\Gamma](P \supset A_0) &= [\Delta](P \supset A_0) && \text{Definition of substitution}\end{aligned}$$

– **Case** $\dfrac{\Gamma \vdash P\ \mathit{prop} \qquad \Gamma \vdash A_0\ \mathit{type}}{\Gamma \vdash A_0 \wedge P\ \mathit{type}}\ \mathsf{WithWF}$

Similar to the ImpliesWF case. □

Lemma 30 (Substitution Invariance)**.**

(i) If $\Gamma \longrightarrow \Delta$ *and* $\Gamma \vdash t : \kappa$ *and* $\mathsf{FEV}([\Gamma]t) = \emptyset$ *then* $[\Delta][\Gamma]t = [\Gamma]t$.

(ii) If $\Gamma \longrightarrow \Delta$ *and* $\Gamma \vdash P\ \mathit{prop}$ *and* $\mathsf{FEV}([\Gamma]P) = \emptyset$ *then* $[\Delta][\Gamma]P = [\Gamma]P$.

(iii) If $\Gamma \longrightarrow \Delta$ *and* $\Gamma \vdash A\ \mathit{type}$ *and* $\mathsf{FEV}([\Gamma]A) = \emptyset$ *then* $[\Delta][\Gamma]A = [\Gamma]A$.

Proof. Each part is a separate induction, relying on the proofs of the earlier parts. In each part, the result follows by an induction on the derivation of $\Gamma \longrightarrow \Delta$.

The main observation is that Δ adds no equations for any variable of t, P, and A that Γ does not already contain, and as a result applying Δ as a substitution to $[\Gamma]t$ does nothing. □

Lemma 24 (Soft Extension)**.**
If $\Gamma \longrightarrow \Delta$ *and* Γ, Θ *ctx and* Θ *is soft, then there exists* Ω *such that* $\mathsf{dom}(\Theta) = \mathsf{dom}(\Omega)$ *and* $\Gamma, \Theta \longrightarrow \Delta, \Omega$.

Proof. By induction on Θ.

- **Case** $\Theta = \cdot$: We have $\Gamma \longrightarrow \Delta$. Let $\Omega = \cdot$. Then $\Gamma, \Theta \longrightarrow \Delta, \Omega$.
- **Case** $\Theta = (\Theta', \hat{\alpha} : \kappa = t)$:

$$\begin{aligned} &\Gamma, \Theta' \longrightarrow \Gamma, \Omega' && \text{By i.h.}\\ \text{☞}\quad &\Gamma, \underbrace{\Theta', \hat{\alpha} : \kappa = t}_{\Theta} \longrightarrow \Delta, \underbrace{\Omega', \hat{\alpha} : \kappa = t}_{\Omega} && \text{By rule } \longrightarrow\mathsf{Solved}\end{aligned}$$

- **Case** $\Theta = (\Theta', \hat{\alpha} : \kappa)$:
  If $\kappa = \star$, let $t = 1$; if $\kappa = \mathbb{N}$, let $t = \mathsf{zero}$.

$$\begin{aligned} &\Gamma, \Theta' \longrightarrow \Gamma, \Omega' && \text{By i.h.}\\ \text{☞}\quad &\Gamma, \underbrace{\Theta', \hat{\alpha} : \kappa}_{\Theta} \longrightarrow \Delta, \underbrace{\Omega', \hat{\alpha} : \kappa = t}_{\Omega} && \text{By rule } \longrightarrow\mathsf{Solve}\end{aligned}$$

□

Lemma 31 (Split Extension)**.**
If $\Delta \longrightarrow \Omega$
and $\hat{\alpha} \in \mathsf{unsolved}(\Delta)$
and $\Omega = \Omega_1[\hat{\alpha} : \kappa = t_1]$
and Ω *is canonical (Definition 3)*
and $\Omega \vdash t_2 : \kappa$
then $\Delta \longrightarrow \Omega_1[\hat{\alpha} : \kappa = t_2]$.

Proof. By induction on the derivation of $\Delta \longrightarrow \Omega$. Use the fact that $\Omega_1[\hat{\alpha} : \kappa = t_1]$ and $\Omega_1[\hat{\alpha} : \kappa = t_2]$ agree on all solutions *except* the solution for $\hat{\alpha}$. In the $\longrightarrow$Solve case where the existential variable is $\hat{\alpha}$, use $\Omega \vdash t_2 : \kappa$. □

D′.1 Reflexivity and Transitivity

Lemma 32 (Extension Reflexivity)**.**
If Γ *ctx then* $\Gamma \longrightarrow \Gamma$*.*

Proof. By induction on the derivation of Γ *ctx*.

- **Case**
$$\frac{}{\cdot\ ctx}\ \mathsf{EmptyCtx}$$
$\cdot \longrightarrow \cdot$ By rule $\longrightarrow$Id

- **Case**
$$\frac{\Gamma\ ctx \qquad x \notin \mathsf{dom}(\Gamma) \qquad \Gamma \vdash A\ type}{\Gamma, x : A\ ctx}\ \mathsf{HypCtx}$$
$\Gamma \longrightarrow \Gamma$ By i.h.
$[\Gamma]A = [\Gamma]A$ By reflexivity
$\Gamma, x : A \longrightarrow \Gamma, x : A$ By rule $\longrightarrow$Var

- **Case**
$$\frac{\Gamma\ ctx \qquad u : \kappa \notin \mathsf{dom}(\Gamma)}{\Gamma, u : \kappa\ ctx}\ \mathsf{VarCtx}$$
$\Gamma \longrightarrow \Gamma$ By i.h.
$\Gamma, u : \kappa \longrightarrow \Gamma, u : \kappa$ By rule $\longrightarrow$Uvar or $\longrightarrow$Unsolved

- **Case**
$$\frac{\Gamma\ ctx \qquad \hat{\alpha} \notin \mathsf{dom}(\Gamma) \qquad \Gamma \vdash t : \kappa}{\Gamma, \hat{\alpha} : \kappa = t\ ctx}\ \mathsf{SolvedCtx}$$
$\Gamma \longrightarrow \Gamma$ By i.h.
$[\Gamma]t = [\Gamma]t$ By reflexivity
$\Gamma, \hat{\alpha} : \kappa = t \longrightarrow \Gamma, \hat{\alpha} : \kappa = t$ By rule $\longrightarrow$Solved

- **Case**
$$\frac{\Gamma\ ctx \qquad \alpha : \kappa \in \Gamma \qquad (\alpha = -) \notin \Gamma \qquad \Gamma \vdash \tau : \kappa}{\Gamma, \alpha = \tau\ ctx}\ \mathsf{EqnVarCtx}$$
$\Gamma \longrightarrow \Gamma$ By i.h.
$[\Gamma]t = [\Gamma]t$ By reflexivity
$\Gamma, \alpha = t \longrightarrow \Gamma, \alpha = t$ By rule $\longrightarrow$Eqn

- **Case**
$$\frac{\Gamma\ ctx \qquad \blacktriangleright_u \notin \Gamma}{\Gamma, \blacktriangleright_u\ ctx}\ \mathsf{MarkerCtx}$$
$\Gamma \longrightarrow \Gamma$ By i.h.
$\Gamma, \blacktriangleright_u \longrightarrow \Gamma, \blacktriangleright_u$ By rule $\longrightarrow$Marker

□

Lemma 33 (Extension Transitivity)**.**
If $\mathcal{D} :: \Gamma \longrightarrow \Theta$ *and* $\mathcal{D}' :: \Theta \longrightarrow \Delta$ *then* $\Gamma \longrightarrow \Delta$*.*

Proof. By induction on $\mathcal{D}'$.

- **Case**
$$\frac{}{\underbrace{\cdot}_{\Theta} \longrightarrow \underbrace{\cdot}_{\Delta}}\ \longrightarrow\mathsf{Id}$$
$\Gamma = \cdot$ By inversion on $\mathcal{D}$
$\cdot \longrightarrow \cdot$ By rule $\longrightarrow$Id
$\Gamma \longrightarrow \Delta$ Since $\Gamma = \Delta = \cdot$

- **Case** $\dfrac{\Theta' \longrightarrow \Delta' \qquad [\Delta']A = [\Delta']A'}{\underbrace{\Theta', x : A}_{\Theta} \longrightarrow \underbrace{\Delta', x : A'}_{\Delta}} \longrightarrow\mathsf{Var}$

  $\Gamma = (\Gamma', x : A'')$ By inversion on $\mathcal{D}$
  $[\Theta]A'' = [\Theta]A$ By inversion on $\mathcal{D}$
  $\Gamma' \longrightarrow \Theta'$ By inversion on $\mathcal{D}$
  $\Gamma' \longrightarrow \Delta'$ By i.h.
  $[\Delta'][\Theta']A'' = [\Delta'][\Theta']A$ By congruence of equality
  $[\Delta']A'' = [\Delta']A$ By Lemma 29 (Substitution Monotonicity)
  $= [\Delta']A'$ By premise $[\Delta']A = [\Delta']A'$
  $\Gamma', x : A'' \longrightarrow \Delta', x : A'$ By $\longrightarrow$Var

- **Case** $\dfrac{\Theta' \longrightarrow \Delta'}{\underbrace{\Theta', \alpha : \kappa}_{\Theta} \longrightarrow \underbrace{\Delta', \alpha : \kappa}_{\Delta}} \longrightarrow\mathsf{Uvar}$

  $\Gamma = (\Gamma', \alpha : \kappa)$ By inversion on $\mathcal{D}$
  $\Gamma' \longrightarrow \Theta'$ By inversion on $\mathcal{D}$
  $\Gamma' \longrightarrow \Delta'$ By i.h.
  $\Gamma', \alpha : \kappa \longrightarrow \Delta', \alpha : \kappa$ By $\longrightarrow$Uvar

- **Case** $\dfrac{\Theta' \longrightarrow \Delta'}{\underbrace{\Theta', \hat{\alpha} : \kappa}_{\Theta} \longrightarrow \underbrace{\Delta', \hat{\alpha} : \kappa}_{\Delta}} \longrightarrow\mathsf{Unsolved}$

  Two rules could have concluded $\mathcal{D} :: \Gamma \longrightarrow (\Theta', \hat{\alpha} : \kappa)$:

  - **Case** $\dfrac{\Gamma' \longrightarrow \Theta'}{\underbrace{\Gamma', \hat{\alpha} : \kappa}_{\Gamma} \longrightarrow \Theta', \hat{\alpha} : \kappa} \longrightarrow\mathsf{Unsolved}$

    $\Gamma' \longrightarrow \Delta'$ By i.h.
    $\Gamma', \hat{\alpha} : \kappa \longrightarrow \Delta', \hat{\alpha} : \kappa$ By rule $\longrightarrow$Add

  - **Case** $\dfrac{\Gamma \longrightarrow \Theta'}{\Gamma \longrightarrow \Theta', \hat{\alpha} : \kappa} \longrightarrow\mathsf{Add}$

    $\Gamma \longrightarrow \Delta'$ By i.h.
    $\Gamma \longrightarrow \Delta', \hat{\alpha} : \kappa$ By rule $\longrightarrow$Add

- **Case** $\dfrac{\Theta' \longrightarrow \Delta' \qquad [\Delta']t = [\Delta']t'}{\underbrace{\Theta', \hat{\alpha} : \kappa = t}_{\Theta} \longrightarrow \underbrace{\Delta', \hat{\alpha} : \kappa = t'}_{\Delta}} \longrightarrow\mathsf{Solved}$

  Two rules could have concluded $\mathcal{D} :: \Gamma \longrightarrow (\Theta', \hat{\alpha} : \kappa = t)$:

  - **Case** $\dfrac{\Gamma' \longrightarrow \Theta' \qquad [\Theta']t'' = [\Theta']t}{\underbrace{\Gamma', \hat{\alpha} : \kappa = t''}_{\Gamma} \longrightarrow \Theta', \hat{\alpha} : \kappa = t} \longrightarrow\mathsf{Solved}$

    $\Gamma' \longrightarrow \Delta'$ By i.h.
    $[\Theta']t'' = [\Theta']t$ Premise
    $[\Delta'][\Theta']t'' = [\Delta'][\Theta']t$ Applying Δ' to both sides
    $[\Delta']t'' = [\Delta']t$ By Lemma 29 (Substitution Monotonicity)
    $= [\Delta']t'$ By premise $[\Delta']t = [\Delta']t'$
    $\Gamma', \hat{\alpha} : \kappa = t'' \longrightarrow \Delta', \hat{\alpha} : \kappa = t'$ By rule $\longrightarrow$Solved

– **Case** $\dfrac{\Gamma \longrightarrow \Theta'}{\Gamma \longrightarrow \Theta', \hat{\alpha} : \kappa = t}\ \longrightarrow\mathsf{AddSolved}$

$\Gamma \longrightarrow \Delta'$ By i.h.
$\Gamma \longrightarrow \Delta', \hat{\alpha} : \kappa = t'$ By rule $\longrightarrow$AddSolved

- **Case** $\dfrac{\Theta' \longrightarrow \Delta' \qquad [\Delta']t = [\Delta']t'}{\underbrace{\Theta', \alpha = t}_{\Theta} \longrightarrow \underbrace{\Delta', \alpha = t'}_{\Delta}}\ \longrightarrow\mathsf{Eqn}$

$\Gamma = (\Gamma', \alpha = t'')$ By inversion on $\mathcal{D}$
$\Gamma' \longrightarrow \Theta'$ By inversion on $\mathcal{D}$
$[\Theta']t'' = [\Theta']t$ By inversion on $\mathcal{D}$
$[\Delta'][\Theta']t'' = [\Delta'][\Theta']t$ Applying Δ' to both sides
$\Gamma' \longrightarrow \Delta'$ By i.h.
$[\Delta']t'' = [\Delta']t$ By Lemma 29 (Substitution Monotonicity)
$= [\Delta']t'$ By premise $[\Delta']t = [\Delta']t'$
$\Gamma', \alpha = t'' \longrightarrow \Delta', \alpha = t'$ By rule $\longrightarrow$Eqn

- **Case** $\dfrac{\Theta \longrightarrow \Delta'}{\Theta \longrightarrow \underbrace{\Delta', \hat{\alpha} : \kappa}_{\Delta}}\ \longrightarrow\mathsf{Add}$

$\Gamma \longrightarrow \Delta'$ By i.h.
$\Gamma \longrightarrow \Delta', \hat{\alpha} : \kappa$ By rule $\longrightarrow$Add

- **Case** $\dfrac{\Theta \longrightarrow \Delta'}{\Theta \longrightarrow \underbrace{\Delta', \hat{\alpha} : \kappa = t}_{\Delta}}\ \longrightarrow\mathsf{AddSolved}$

$\Gamma \longrightarrow \Delta'$ By i.h.
$\Gamma \longrightarrow \Delta', \hat{\alpha} : \kappa = t$ By rule $\longrightarrow$AddSolved

- **Case** $\dfrac{\Theta' \longrightarrow \Delta'}{\underbrace{\Theta', \blacktriangleright_u}_{\Theta} \longrightarrow \underbrace{\Delta', \blacktriangleright_u}_{\Delta}}\ \longrightarrow\mathsf{Marker}$

$\Gamma = \Gamma', \blacktriangleright_u$ By inversion on $\mathcal{D}$
$\Gamma' \longrightarrow \Theta'$ By inversion on $\mathcal{D}$
$\Gamma' \longrightarrow \Delta'$ By i.h.
$\Gamma', \blacktriangleright_u \longrightarrow \Delta', \blacktriangleright_u$ By $\longrightarrow$Uvar □

D′.2 Weakening

Lemma 34 (Suffix Weakening)**.** *If* $\Gamma \vdash t : \kappa$ *then* $\Gamma, \Theta \vdash t : \kappa$.

Proof. By induction on the given derivation. All cases are straightforward. □

Lemma 35 (Suffix Weakening)**.** *If* $\Gamma \vdash A$ *type then* $\Gamma, \Theta \vdash A$ *type*.

Proof. By induction on the given derivation. All cases are straightforward. □

Lemma 36 (Extension Weakening (Sorts))**.** *If* $\Gamma \vdash t : \kappa$ *and* $\Gamma \longrightarrow \Delta$ *then* $\Delta \vdash t : \kappa$.

Proof. By a straightforward induction on $\Gamma \vdash t : \kappa$.

In the VarSort case, use Lemma 22 (Extension Inversion) (i) or (v). In the SolvedVarSort case, use Lemma 22 (Extension Inversion) (iv). In the other cases, apply the i.h. to all subderivations, then apply the rule. □

Lemma 37 (Extension Weakening (Props))**.** *If* $\Gamma \vdash P$ *prop and* $\Gamma \longrightarrow \Delta$ *then* $\Delta \vdash P$ *prop.*

Proof. By inversion on rule EqProp, and Lemma 36 (Extension Weakening (Sorts)) twice. □

Lemma 38 (Extension Weakening (Types))**.** *If* $\Gamma \vdash A$ *type and* $\Gamma \longrightarrow \Delta$ *then* $\Delta \vdash A$ *type.*

Proof. By a straightforward induction on $\Gamma \vdash A$ *type*.

In the VarWF case, use Lemma 22 (Extension Inversion) (i) or (v). In the SolvedVarWF case, use Lemma 22 (Extension Inversion) (iv).

In the other cases, apply the i.h. and/or (for ImpliesWF and WithWF) Lemma 37 (Extension Weakening (Props)) to all subderivations, then apply the rule. □

D′.3 Principal Typing Properties

Lemma 39 (Principal Agreement)**.**

(i) If $\Gamma \vdash A$ **!** *type and* $\Gamma \longrightarrow \Delta$ *then* $[\Delta]A = [\Gamma]A$*.*

(ii) If $\Gamma \vdash P$ *prop and* $\mathsf{FEV}(P) = \emptyset$ *and* $\Gamma \longrightarrow \Delta$ *then* $[\Delta]P = [\Gamma]P$*.*

Proof. By induction on the derivation of $\Gamma \longrightarrow \Delta$.

Part (i):

- **Case** $\dfrac{\Gamma_0 \longrightarrow \Delta_0 \qquad [\Delta_0]t = [\Delta_0]t'}{\Gamma_0, \alpha = t \longrightarrow \underbrace{\Delta_0, \alpha = t'}_{\Delta}} \longrightarrow\mathsf{Eqn}$

  If $\alpha \notin \mathrm{FV}(A)$, then:

  $$\begin{aligned}[\Gamma_0, \alpha = t]A &= [\Gamma_0]A && \text{By def. of subst.}\\ &= [\Delta_0]A && \text{By i.h.}\\ &= [\Delta_0, \alpha = t']A && \text{By def. of subst.}\end{aligned}$$

  Otherwise, $\alpha \in \mathrm{FV}(A)$.

  $\Gamma_0 \vdash t$ *type* Γ is well-formed
  $\Gamma_0 \vdash [\Gamma_0]t$ *type* By Lemma 13 (Right-Hand Substitution for Typing)

  Suppose, for a contradiction, that $\mathsf{FEV}([\Gamma_0]t) \neq \emptyset$.

  Since $\alpha \in \mathrm{FV}(A)$, we also have $\mathsf{FEV}([\Gamma]A) \neq \emptyset$, a contradiction.

  $$\begin{aligned}\mathsf{FEV}([\Gamma_0]t) &\neq \emptyset && \text{Assumption (for contradiction)}\\ [\Gamma_0]t &= [\Gamma]\alpha && \text{By def. of subst.}\\ \mathsf{FEV}([\Gamma]\alpha) &\neq \emptyset && \text{By above equality}\\ \alpha &\in \mathrm{FV}(A) && \text{Above}\\ \mathsf{FEV}([\Gamma]A) &\neq \emptyset && \text{By a property of subst.}\\ \Gamma &\vdash A \text{ ! } \textit{type} && \text{Given}\\ \mathsf{FEV}([\Gamma]A) &= \emptyset && \text{By inversion}\\ &\Rightarrow\Leftarrow\\ \mathsf{FEV}([\Gamma_0]t) &= \emptyset && \text{By contradiction}\\ \Gamma_0 &\vdash t \text{ ! } \textit{type} && \text{By PrincipalWF}\\ [\Gamma_0]t &= [\Delta_0]t && \text{By i.h.}\end{aligned}$$

  $$\begin{aligned}\Gamma_0 &\vdash [\Delta_0]t \textit{ type} && \text{By above equality}\\ \mathsf{FEV}([\Delta_0]t) &= \emptyset && \text{By above equality}\\ \Gamma_0 &\vdash \left[[\Delta_0]t/\alpha\right]A \text{ ! } \textit{type} && \text{By Lemma 8 (Substitution—Well-formedness) (i)}\\ [\Gamma_0]\left[[\Delta_0]t/\alpha\right]A &= [\Delta_0]\left[[\Delta_0]t/\alpha\right]A && \text{By i.h. (at } \left[[\Delta_0]t/\alpha\right]A)\end{aligned}$$

  $$\begin{aligned}[\Gamma_0, \alpha = t]A &= \left[\Gamma_0\right]\left[[\Gamma_0]t/\alpha\right]A && \text{By def. of subst.}\\ &= \left[\Gamma_0\right]\left[[\Delta_0]t/\alpha\right]A && \text{By above equality}\\ &= \left[\Delta_0\right]\left[[\Delta_0]t/\alpha\right]A && \text{By above equality}\\ &= \left[\Delta_0\right]\left[[\Delta_0]t'/\alpha\right]A && \text{By } [\Delta_0]t = [\Delta_0]t'\\ &= [\Delta]A && \text{By def. of subst.}\end{aligned}$$

- **Case** $\longrightarrow$Solved, $\longrightarrow$Solve, $\longrightarrow$Add, $\longrightarrow$Solved: Similar to the $\longrightarrow$Eqn case.
- **Case** $\longrightarrow$Id, $\longrightarrow$Var, $\longrightarrow$Uvar, $\longrightarrow$Unsolved, $\longrightarrow$Marker:
  Straightforward, using the i.h. and the definition of substitution.

Part (ii): Similar to part (i), using part (ii) of Lemma 8 (Substitution—Well-formedness). □

Lemma 40 (Right-Hand Subst. for Principal Typing)**.** *If* $\Gamma \vdash A\ \mathfrak{p}$ *type then* $\Gamma \vdash [\Gamma]A\ \mathfrak{p}$ *type*.

Proof. By cases of $\mathfrak{p}$:

- Case $\mathfrak{p} = !$:

$\Gamma \vdash A$ *type*	By inversion
$\mathsf{FEV}([\Gamma]A) = \emptyset$	By inversion
$\Gamma \vdash [\Gamma]A$ *type*	By Lemma 13 (Right-Hand Substitution for Typing)
$\Gamma \longrightarrow \Gamma$	By Lemma 32 (Extension Reflexivity)
$[\Gamma][\Gamma]A = [\Gamma]A$	By Lemma 29 (Substitution Monotonicity)
$\mathsf{FEV}([\Gamma][\Gamma]A) = \emptyset$	By inversion
$\Gamma \vdash [\Gamma]A\ !$ *type*	By rule PrincipalWF

- Case $\mathfrak{p} = \not!$:

$\Gamma \vdash A$ *type*	By inversion
$\Gamma \vdash [\Gamma]A$ *type*	By Lemma 13 (Right-Hand Substitution for Typing)
$\Gamma \vdash A \not!$ *type*	By rule NonPrincipalWF

□

Lemma 41 (Extension Weakening for Principal Typing)**.** *If* $\Gamma \vdash A\ \mathfrak{p}$ *type and* $\Gamma \longrightarrow \Delta$ *then* $\Delta \vdash A\ \mathfrak{p}$ *type.*

Proof. By cases of $\mathfrak{p}$:

- Case $\mathfrak{p} = \not!$:

$\Gamma \vdash A$ *type*	By inversion
$\Delta \vdash A$ *type*	By Lemma 38 (Extension Weakening (Types))
$\Delta \vdash A \not!$ *type*	By rule NonPrincipalWF

- Case $\mathfrak{p} = !$:

$\Gamma \vdash A$ *type*	By inversion
$\mathsf{FEV}([\Gamma]A) = \emptyset$	By inversion
$\Delta \vdash A$ *type*	By Lemma 38 (Extension Weakening (Types))
$\Delta \vdash [\Delta]A$ *type*	By Lemma 13 (Right-Hand Substitution for Typing)
$[\Delta]A = [\Gamma]A$	By Lemma 30 (Substitution Invariance)
$\mathsf{FEV}([\Delta]A) = \emptyset$	By congruence of equality
$\Delta \vdash [\Delta]A\ !$ *type*	By rule PrincipalWF

□

Lemma 42 (Inversion of Principal Typing)**.**

(1) If $\Gamma \vdash (A \to B)\ \mathfrak{p}$ *type then* $\Gamma \vdash A\ \mathfrak{p}$ *type and* $\Gamma \vdash B\ \mathfrak{p}$ *type.*

(2) If $\Gamma \vdash (P \supset A)\ \mathfrak{p}$ *type then* $\Gamma \vdash P$ *prop and* $\Gamma \vdash A\ \mathfrak{p}$ *type.*

(3) If $\Gamma \vdash (A \wedge P)\ \mathfrak{p}$ *type then* $\Gamma \vdash P$ *prop and* $\Gamma \vdash A\ \mathfrak{p}$ *type.*

Proof. Proof of part 1:
We have $\Gamma \vdash A \to B\ \mathfrak{p}$ *type*.

- Case $\mathfrak{p} = \not!$:

1	$\Gamma \vdash A \to B$ *type*	By inversion
	$\Gamma \vdash A$ *type*	By inversion on 1
	$\Gamma \vdash B$ *type*	By inversion on 1
	$\Gamma \vdash A \not!$ *type*	By rule NonPrincipalWF
	$\Gamma \vdash B \not!$ *type*	By rule NonPrincipalWF

- Case $p = !$:

1	$\Gamma \vdash A \to B$ *type*	By inversion on $\Gamma \vdash A \to B\ !$ *type*
	$\emptyset = \mathsf{FEV}([\Gamma](A \to B))$	$''$
	$= \mathsf{FEV}([\Gamma]A \to [\Gamma]B)$	By definition of substitution
	$= \mathsf{FEV}([\Gamma]A) \cup \mathsf{FEV}([\Gamma]B)$	By definition of $\mathsf{FEV}(-)$
	$\mathsf{FEV}([\Gamma]A) = \mathsf{FEV}([\Gamma]B) = \emptyset$	By properties of empty sets and unions
	$\Gamma \vdash A$ *type*	By inversion on 1
	$\Gamma \vdash B$ *type*	By inversion on 1
	$\Gamma \vdash A\ !$ *type*	By rule PrincipalWF
	$\Gamma \vdash B\ !$ *type*	By rule PrincipalWF

Part 2: We have $\Gamma \vdash P \supset A\ p$ *type*. Similar to Part 1.
Part 3: We have $\Gamma \vdash A \wedge P\ p$ *type*. Similar to Part 2. □

D′.4 Instantiation Extends

Lemma 43 (Instantiation Extension)**.**
If $\Gamma \vdash \hat{\alpha} := \tau : \kappa \dashv \Delta$ *then* $\Gamma \longrightarrow \Delta$.

Proof. By induction on the given derivation.

- **Case**
$$\frac{\Gamma_L \vdash \tau : \kappa}{\underbrace{\Gamma_L, \hat{\alpha} : \kappa, \Gamma_R}_{\Gamma} \vdash \hat{\alpha} := \tau : \kappa \dashv \Gamma_L, \hat{\alpha} : \kappa = \tau, \Gamma_R}\ \mathsf{InstSolve}$$
Follows by Lemma 23 (Deep Evar Introduction) (ii).

- **Case**
$$\frac{\hat{\beta} \in \mathsf{unsolved}(\Gamma_0[\hat{\alpha} : \kappa][\hat{\beta} : \kappa])}{\underbrace{\Gamma_0[\hat{\alpha} : \kappa][\hat{\beta} : \kappa]}_{\Gamma} \vdash \hat{\alpha} := \hat{\beta} : \kappa \dashv \Gamma_0[\hat{\alpha} : \kappa][\hat{\beta} : \kappa = \hat{\alpha}]}\ \mathsf{InstReach}$$
Follows by Lemma 23 (Deep Evar Introduction) (ii).

- **Case**
$$\frac{\Gamma_0[\hat{\alpha}_2 : \star, \hat{\alpha}_1 : \star, \hat{\alpha} : \star = \hat{\alpha}_1 \oplus \hat{\alpha}_2] \vdash \hat{\alpha}_1 := \tau_1 : \star \dashv \Theta \qquad \Theta \vdash \hat{\alpha}_2 := [\Theta]\tau_2 : \star \dashv \Delta}{\Gamma_0[\hat{\alpha} : \star] \vdash \hat{\alpha} := \tau_1 \oplus \tau_2 : \star \dashv \Delta}\ \mathsf{InstBin}$$

$\Gamma_0[\hat{\alpha}_2 : \star, \hat{\alpha}_1 : \star, \hat{\alpha} : \star = \hat{\alpha}_1 \oplus \hat{\alpha}_2] \vdash \hat{\alpha}_1 := \tau_1 : \star \dashv \Theta$	Subderivation
$\Gamma_0[\hat{\alpha}_2 : \star, \hat{\alpha}_1 : \star, \hat{\alpha} : \star = \hat{\alpha}_1 \oplus \hat{\alpha}_2] \longrightarrow \Theta$	By i.h.
$\Theta \vdash \hat{\alpha}_2 := [\Theta]\tau_2 : \star \dashv \Delta$	Subderivation
$\Theta \longrightarrow \Delta$	By i.h.
$\Gamma_0[\hat{\alpha}_2 : \star, \hat{\alpha}_1 : \star, \hat{\alpha} : \star = \hat{\alpha}_1 \oplus \hat{\alpha}_2] \longrightarrow \Delta$	By Lemma 33 (Extension Transitivity)
$\Gamma_0[\hat{\alpha} : \star] \longrightarrow \Gamma_0[\hat{\alpha}_2 : \star, \hat{\alpha}_1 : \star, \hat{\alpha} : \star = \hat{\alpha}_1 \oplus \hat{\alpha}_2]$	By Lemma 23 (Deep Evar Introduction) (parts (i), (i), and (ii), using Lemma 33 (Extension Transitivity))
$\Gamma_0[\hat{\alpha} : \star] \longrightarrow \Delta$	By Lemma 33 (Extension Transitivity)

- **Case**
$$\frac{}{\Gamma_0[\hat{\alpha} : \mathbb{N}] \vdash \hat{\alpha} := \mathsf{zero} : \mathbb{N} \dashv \Gamma_0[\hat{\alpha} : \mathbb{N} = \mathsf{zero}]}\ \mathsf{InstZero}$$
Follows by Lemma 23 (Deep Evar Introduction) (ii).

- **Case**
$$\frac{\Gamma[\hat{\alpha}_1 : \mathbb{N}, \hat{\alpha} : \mathbb{N} = \mathsf{succ}(\hat{\alpha}_1)] \vdash \hat{\alpha}_1 := t_1 : \mathbb{N} \dashv \Delta}{\Gamma[\hat{\alpha} : \mathbb{N}] \vdash \hat{\alpha} := \mathsf{succ}(t_1) : \mathbb{N} \dashv \Delta}\ \mathsf{InstSucc}$$
By reasoning similar to the InstBin case. □

D′.5 Equivalence Extends

Lemma 44 (Elimeq Extension)**.**
If $\Gamma \mathrel{/} s \mathrel{\hat{=}} t : \kappa \dashv \Delta$ *then there exists* Θ *such that* $\Gamma, \Theta \longrightarrow \Delta$*.*

Proof. By induction on the given derivation. Note that the statement restricts the output to be a (consistent) context Δ.

- **Case**
$$\frac{}{\Gamma \mathrel{/} \alpha \mathrel{\hat{=}} \alpha : \kappa \dashv \Gamma}\ \mathsf{ElimeqUvarRefl}$$
Since $\Delta = \Gamma$, applying Lemma 32 (Extension Reflexivity) suffices (let $\Theta = \cdot$).

- **Case**
$$\frac{}{\Gamma \mathrel{/} \mathsf{zero} \mathrel{\hat{=}} \mathsf{zero} : \mathbb{N} \dashv \Gamma}\ \mathsf{ElimeqZero}$$
Similar to the ElimeqUvarRefl case.

- **Case**
$$\frac{\Gamma \mathrel{/} \sigma \mathrel{\hat{=}} t : \mathbb{N} \dashv \Delta}{\Gamma \mathrel{/} \mathsf{succ}(\sigma) \mathrel{\hat{=}} \mathsf{succ}(t) : \mathbb{N} \dashv \Delta}\ \mathsf{ElimeqSucc}$$
Follows by i.h.

- **Case**
$$\frac{\Gamma_0[\hat{\alpha} : \kappa] \vdash \hat{\alpha} := t : \kappa \dashv \Delta}{\underbrace{\Gamma_0[\hat{\alpha} : \kappa]}_{\Gamma} \mathrel{/} \hat{\alpha} \mathrel{\hat{=}} t : \kappa \dashv \Delta}\ \mathsf{ElimeqInstL}$$
$\Gamma \vdash \hat{\alpha} := t : \kappa \dashv \Delta$ Subderivation
$\Gamma \longrightarrow \Delta$ By Lemma 43 (Instantiation Extension)
Let $\Theta = \cdot$.
☞ $\Gamma, \Theta \longrightarrow \Delta$ By $\Theta = \cdot$

- **Case**
$$\frac{\alpha \notin \mathrm{FV}([\Gamma]t) \qquad (\alpha = -) \notin \Gamma}{\Gamma \mathrel{/} \alpha \mathrel{\hat{=}} t : \kappa \dashv \Gamma, \alpha = t}\ \mathsf{ElimeqUvarL}$$
Let Θ be $(\alpha = t)$.
☞ $\Gamma, \underbrace{\alpha = t}_{\Theta} \longrightarrow \Gamma, \alpha = t$ By Lemma 32 (Extension Reflexivity)

- **Cases** ElimeqInstR, ElimeqUvarR:
Similar to the respective L cases.

- **Case**
$$\frac{\sigma \# t}{\Gamma \mathrel{/} \sigma \mathrel{\hat{=}} t : \kappa \dashv \bot}\ \mathsf{ElimeqClash}$$
The statement says that the output is a (consistent) context Δ, so this case is impossible. □

Lemma 45 (Elimprop Extension)**.**
If $\Gamma \mathrel{/} P \dashv \Delta$ *then there exists* Θ *such that* $\Gamma, \Theta \longrightarrow \Delta$*.*

Proof. By induction on the given derivation. Note that the statement restricts the output to be a (consistent) context Δ.

- **Case**
$$\frac{\Gamma \mathrel{/} \sigma \mathrel{\hat{=}} t : \mathbb{N} \dashv \Delta}{\Gamma \mathrel{/} \sigma = t \dashv \Delta}\ \mathsf{ElimpropEq}$$
$\Gamma \mathrel{/} \sigma \mathrel{\hat{=}} t : \mathbb{N} \dashv \Delta$ Subderivation
☞ $\Gamma, \Theta \longrightarrow \Delta$ By Lemma 44 (Elimeq Extension) □

Lemma 46 (Checkeq Extension)**.**
If $\Gamma \vdash A \equiv B \dashv \Delta$ *then* $\Gamma \longrightarrow \Delta$.

Proof. By induction on the given derivation.

- **Case**
$$\frac{}{\Gamma \vdash u \doteq u : \kappa \dashv \Gamma}\ \mathsf{CheckeqVar}$$
Since $\Delta = \Gamma$, applying Lemma 32 (Extension Reflexivity) suffices.

- **Cases** CheckeqUnit, CheckeqZero: Similar to the CheckeqVar case.

- **Case**
$$\frac{\Gamma \vdash \tau_1 \doteq \tau_1' : \star \dashv \Theta \qquad \Theta \vdash [\Theta]\tau_2 \doteq [\Theta]\tau_2' : \star \dashv \Delta}{\Gamma \vdash \tau_1 \oplus \tau_2 \doteq \tau_1' \oplus \tau_2' : \star \dashv \Delta}\ \mathsf{CheckeqBin}$$
$\Gamma \longrightarrow \Theta$ By i.h.
$\Theta \longrightarrow \Delta$ By i.h.
☞ $\Gamma \longrightarrow \Delta$ By Lemma 33 (Extension Transitivity)

- **Case**
$$\frac{\Gamma \vdash \sigma \doteq t : \mathbb{N} \dashv \Delta}{\Gamma \vdash \mathsf{succ}(\sigma) \doteq \mathsf{succ}(t) : \mathbb{N} \dashv \Delta}\ \mathsf{CheckeqSucc}$$
$\Gamma \vdash \sigma \doteq t : \mathbb{N} \dashv \Delta$ Subderivation
☞ $\Gamma \longrightarrow \Delta$ By i.h.

- **Case**
$$\frac{\Gamma_0[\hat{\alpha}] \vdash \hat{\alpha} := t : \kappa \dashv \Delta \qquad \hat{\alpha} \notin \mathrm{FV}([\Gamma_0[\hat{\alpha}]]t)}{\Gamma_0[\hat{\alpha}] \vdash \hat{\alpha} \doteq t : \kappa \dashv \Delta}\ \mathsf{CheckeqInstL}$$
$\Gamma_0[\hat{\alpha}] \vdash \hat{\alpha} := t : \kappa \dashv \Delta$ Subderivation
☞ $\underbrace{\Gamma_0[\hat{\alpha}]}_{\Gamma} \longrightarrow \Delta$ By Lemma 43 (Instantiation Extension)

- **Case** CheckeqInstR: Similar to the CheckeqInstL case. □

Lemma 47 (Checkprop Extension)**.**
If $\Gamma \vdash P\ \mathit{true} \dashv \Delta$ *then* $\Gamma \longrightarrow \Delta$.

Proof. By induction on the given derivation.

- **Case**
$$\frac{\Gamma \vdash \sigma \doteq t : \mathbb{N} \dashv \Delta}{\Gamma \vdash \sigma = t\ \mathit{true} \dashv \Delta}\ \mathsf{CheckpropEq}$$
$\Gamma \vdash \sigma \doteq t : \mathbb{N} \dashv \Delta$ Subderivation
☞ $\Gamma \longrightarrow \Delta$ By Lemma 46 (Checkeq Extension) □

Lemma 48 (Prop Equivalence Extension)**.**
If $\Gamma \vdash P \equiv Q \dashv \Delta$ *then* $\Gamma \longrightarrow \Delta$.

Proof. By induction on the given derivation.

- **Case**
$$\frac{\Gamma \vdash \sigma_1 \doteq \tau_1 : \mathbb{N} \dashv \Theta \qquad \Theta \vdash \sigma_2 \doteq \tau_2 : \mathbb{N} \dashv \Delta}{\Gamma \vdash (\sigma_1 = \sigma_2) \equiv (\tau_1 = \tau_2) \dashv \Delta}\ \equiv\mathsf{PropEq}$$
$\Gamma \vdash \sigma_1 \doteq \tau_1 : \mathbb{N} \dashv \Theta$ Subderivation
$\Gamma \longrightarrow \Theta$ By Lemma 46 (Checkeq Extension)
$\Theta \vdash \sigma_2 \doteq \tau_2 : \mathbb{N} \dashv \Delta$ Subderivation
$\Theta \longrightarrow \Delta$ By Lemma 46 (Checkeq Extension)
☞ $\Gamma \longrightarrow \Delta$ By Lemma 33 (Extension Transitivity) □

Lemma 49 (Equivalence Extension)**.**
If $\Gamma \vdash A \equiv B \dashv \Delta$ *then* $\Gamma \longrightarrow \Delta$.

Proof. By induction on the given derivation.

- **Case**
$$\frac{}{\Gamma \vdash \alpha \equiv \alpha \dashv \Gamma}\ \equiv\mathsf{Var}$$
Here $\Delta = \Gamma$, so Lemma 32 (Extension Reflexivity) suffices.

- **Case**
$$\frac{}{\Gamma \vdash \hat{\alpha} \equiv \hat{\alpha} \dashv \Gamma}\ \equiv\mathsf{Exvar}$$
Similar to the $\equiv$Var case.

- **Case**
$$\frac{}{\Gamma \vdash 1 \equiv 1 \dashv \Gamma}\ \equiv\mathsf{Unit}$$
Similar to the $\equiv$Var case.

- **Case**
$$\frac{\Gamma \vdash A_1 \equiv B_1 \dashv \Theta \qquad \Theta \vdash [\Theta]A_2 \equiv [\Theta]B_2 \dashv \Delta}{\Gamma \vdash (A_1 \oplus A_2) \equiv (B_1 \oplus B_2) \dashv \Delta}\ \equiv\oplus$$

	$\Gamma \vdash A_1 \equiv B_1 \dashv \Theta$	Subderivation
	$\Gamma \longrightarrow \Theta$	By i.h.
	$\Theta \vdash [\Theta]A_2 \equiv [\Theta]B_2 \dashv \Delta$	Subderivation
	$\Theta \longrightarrow \Delta$	By i.h.
☞	$\Gamma \longrightarrow \Delta$	By Lemma 33 (Extension Transitivity)

- **Case** $\equiv$Vec: Similar to the $\equiv\oplus$ case.

- **Cases** $\equiv\supset$, $\equiv\wedge$: Similar to the $\equiv\oplus$ case, but with Lemma 48 (Prop Equivalence Extension) on the first premise.

- **Case**
$$\frac{\Gamma, \alpha : \kappa \vdash A_0 \equiv B \dashv \Delta, \alpha : \kappa, \Delta'}{\Gamma \vdash \forall \alpha : \kappa.\, A_0 \equiv \forall \alpha : \kappa.\, B \dashv \Delta}\ \equiv\forall$$

	$\Gamma, \alpha : \kappa \vdash A_0 \equiv B \dashv \Delta, \alpha : \kappa, \Delta'$	Subderivation
	$\Gamma, \alpha : \kappa \longrightarrow \Delta, \alpha : \kappa, \Delta'$	By i.h.
☞	$\Gamma \longrightarrow \Delta$	By Lemma 22 (Extension Inversion) (i)

- **Case**
$$\frac{\Gamma_0[\hat{\alpha}] \vdash \hat{\alpha} := \tau : \star \dashv \Delta \qquad \hat{\alpha} \notin \mathrm{FV}([\Gamma_0[\hat{\alpha}]]\tau)}{\Gamma_0[\hat{\alpha}] \vdash \hat{\alpha} \equiv \tau \dashv \Delta}\ \equiv\mathsf{InstantiateL}$$

	$\Gamma_0[\hat{\alpha}] \vdash \hat{\alpha} := \tau : \star \dashv \Delta$	Subderivation
☞	$\underbrace{\Gamma_0[\hat{\alpha}]}_{\Gamma} \longrightarrow \Delta$	By Lemma 43 (Instantiation Extension)

- **Case** $\equiv$InstantiateR: Similar to the $\equiv$InstantiateL case. □

D′.6 Subtyping Extends

Lemma 50 (Subtyping Extension)**.** *If* $\Gamma \vdash A <:^{\mp} B \dashv \Delta$ *then* $\Gamma \longrightarrow \Delta$.

Proof. By induction on the given derivation.

- **Case** $\dfrac{\Gamma, \blacktriangleright_{\hat{\alpha}}, \hat{\alpha} : \kappa \vdash [\hat{\alpha}/\alpha]A <:^{-} B \dashv \Delta, \blacktriangleright_{\hat{\alpha}}, \Theta}{\Gamma \vdash \forall \alpha : \kappa.\, A <:^{-} B \dashv \Delta}\, <:\forall L$

	$\Gamma, \blacktriangleright_{\hat{\alpha}}, \hat{\alpha} : \kappa \vdash [\hat{\alpha}/\alpha]A <:^{-} B \dashv \Delta, \blacktriangleright_{\hat{\alpha}}, \Theta$	Subderivation
	$\Gamma, \blacktriangleright_{\hat{\alpha}}, \hat{\alpha} : \kappa \longrightarrow \Delta, \blacktriangleright_{\hat{\alpha}}, \Theta$	By i.h. (i)
☞	$\Gamma \longrightarrow \Delta$	By Lemma 22 (Extension Inversion) (ii)

- **Case** <:∃R: Similar to the <:∀L case.

- **Case** $\dfrac{\Gamma, \alpha : \kappa \vdash A <:^{*} B \dashv \Delta, \alpha : \kappa, \Theta}{\Gamma \vdash A <:^{*} \forall \alpha : \kappa.\, B \dashv \Delta}\, <:\forall R$

  Similar to the <:∀L case, but using part (i) of Lemma 22 (Extension Inversion).

- **Case** <:∃L: Similar to the <:∀R case.

- **Case** $\dfrac{\Gamma \vdash A \equiv B \dashv \Delta}{\Gamma \vdash A <:^{*} B \dashv \Delta}\, <:\mathsf{Equiv}$

	$\Gamma \vdash A \equiv B \dashv \Delta$	Subderivation
☞	$\Gamma \longrightarrow \Delta$	By Lemma 49 (Equivalence Extension)

□

D′.7 Typing Extends

Lemma 51 (Typing Extension)**.**
If $\Gamma \vdash e \Leftarrow A\ p \dashv \Delta$
or $\Gamma \vdash e \Rightarrow A\ p \dashv \Delta$
or $\Gamma \vdash s : A\ p \gg B\ q \dashv \Delta$
or $\Gamma \vdash \Pi :: \vec{A} \Leftarrow C\ p \dashv \Delta$
or $\Gamma\ /\ P \vdash \Pi :: \vec{A} \Leftarrow C\ p \dashv \Delta$
then $\Gamma \longrightarrow \Delta$.

Proof. By induction on the given derivation.

- **Match judgments:**

  In rule MatchEmpty, $\Delta = \Gamma$, so the result follows by Lemma 32 (Extension Reflexivity).

  Rules MatchBase, Match×, Match$+_k$ and MatchWild each have a single premise in which the contexts match the conclusion (input Γ and output Δ), so the result follows by i.h. For rule MatchSeq, Lemma 33 (Extension Transitivity) is also needed.

  In rule Match∃, apply the i.h., then use Lemma 22 (Extension Inversion) (i).

  Match∧: Use the i.h.

  MatchNeg: Use the i.h. and Lemma 22 (Extension Inversion) (v).

  Match⊥: Immediate by Lemma 32 (Extension Reflexivity).

  MatchUnify:

	$\Gamma, \blacktriangleright_P, \Theta' \longrightarrow \Theta$	By Lemma 44 (Elimeq Extension)
	$\Theta \longrightarrow \Delta, \blacktriangleright_P, \Delta'$	By i.h.
	$\Gamma, \blacktriangleright_P, \Theta' \longrightarrow \Delta, \blacktriangleright_P, \Delta'$	By Lemma 33 (Extension Transitivity)
☞	$\Gamma \longrightarrow \Delta$	By Lemma 22 (Extension Inversion) (ii)

- **Synthesis, checking, and spine judgments:** In rules Var, 1I, EmptySpine, and ⊃I⊥, the output context Δ is exactly Γ, so the result follows by Lemma 32 (Extension Reflexivity).

  - **Case** ∀I: Use the i.h. and Lemma 33 (Extension Transitivity).
  - **Case** ∀Spine: By ⟶Add, $\Gamma \longrightarrow \Gamma, \hat{\alpha} : \kappa$.
    The result follows by i.h. and Lemma 33 (Extension Transitivity).

- **Cases** $\wedge$I, $\supset$Spine: Use Lemma 47 (Checkprop Extension), the i.h., and Lemma 33 (Extension Transitivity).
- **Cases** Nil, Cons: Using reasoning found in the $\wedge$I and $\supset$I cases.
- **Case** $\exists$I: Use the i.h.
- **Case** $\supset$I:

$$\begin{array}{rll} \Gamma, \blacktriangleright_P, \Theta' \longrightarrow \Theta & & \text{By Lemma 45 (Elimprop Extension)} \\ \Theta \longrightarrow \Delta, \blacktriangleright_P, \Delta & & \text{By i.h.} \\ \Gamma, \blacktriangleright_P, \Theta' \longrightarrow \Delta, \blacktriangleright_P, \Delta & & \text{By Lemma 33 (Extension Transitivity)} \\ \Gamma \longrightarrow \Delta & \text{☞} & \text{By Lemma 22 (Extension Inversion)} \end{array}$$

- **Cases** $\rightarrow$I, Rec: Use the i.h. and Lemma 22 (Extension Inversion).
- **Cases** Sub, Anno, $\rightarrow$E, $\rightarrow$E-!, $\rightarrow$Spine, $+\mathsf{I}_k$, $\times$I:
  Use the i.h., and Lemma 33 (Extension Transitivity) as needed.
- **Case** 1I$\hat{\alpha}$: By Lemma 23 (Deep Evar Introduction) (ii).
- **Case** $\hat{\alpha}$Spine, $+\mathsf{I}\hat{\alpha}_k$, $\times\mathsf{I}\hat{\alpha}$:
  Use Lemma 23 (Deep Evar Introduction) (i) twice, Lemma 23 (Deep Evar Introduction) (ii), the i.h., and Lemma 33 (Extension Transitivity).
- **Case** $\rightarrow\mathsf{I}\hat{\alpha}$: Use Lemma 23 (Deep Evar Introduction) (i) twice, Lemma 23 (Deep Evar Introduction) (ii), the i.h. and Lemma 22 (Extension Inversion) (v).
- **Case** Case: Use the i.h. on the synthesis premise and the match premise, and then Lemma 33 (Extension Transitivity). □

D′.8 Unfiled

Lemma 52 (Context Partitioning)**.**
If $\Delta, \blacktriangleright_{\hat{\alpha}}, \Theta \longrightarrow \Omega, \blacktriangleright_{\hat{\alpha}}, \Omega_Z$ *then there is a* Ψ *such that* $[\Omega, \blacktriangleright_{\hat{\alpha}}, \Omega_Z](\Delta, \blacktriangleright_{\hat{\alpha}}, \Theta) = [\Omega]\Delta, \Psi$.

Proof. By induction on the given derivation.

- **Case** $\longrightarrow$Id: Impossible: $\Delta, \blacktriangleright_{\hat{\alpha}}, \Theta$ cannot have the form $\cdot$.
- **Case** $\longrightarrow$Var: We have $\Omega_Z = (\Omega'_Z, x : A)$ and $\Theta = (\Theta', x : A')$. By i.h., there is Ψ' such that $[\Omega, \blacktriangleright_{\hat{\alpha}}, \Omega'_Z](\Delta, \blacktriangleright_{\hat{\alpha}}, \Theta') = [\Omega]\Delta, \Psi'$. Then by the definition of context application, $[\Omega, \blacktriangleright_{\hat{\alpha}}, \Omega'_Z, x : A](\Delta, \blacktriangleright_{\hat{\alpha}}, \Theta', x : A') = [\Omega]\Delta, \Psi', x : [\Omega']A$. Let $\Psi = (\Psi', x : [\Omega']A)$.
- **Case** $\longrightarrow$Uvar: Similar to the $\longrightarrow$Var case, with $\Psi = (\Psi', \alpha : \kappa)$.
- **Cases** $\longrightarrow$Eqn, $\longrightarrow$Unsolved, $\longrightarrow$Solved, $\longrightarrow$Solve, $\longrightarrow$Add, $\longrightarrow$AddSolved, $\longrightarrow$Marker:
  Broadly similar to the $\longrightarrow$Uvar case, but the rightmost context element disappears in context application, so we let $\Psi = \Psi'$. □

Lemma 54 (Completing Stability)**.**
If $\Gamma \longrightarrow \Omega$ *then* $[\Omega]\Gamma = [\Omega]\Omega$.

Proof. By induction on the derivation of $\Gamma \longrightarrow \Omega$.

- **Case**

$$\frac{}{\cdot \longrightarrow \cdot}\ \longrightarrow\mathsf{Id}$$

  Immediate.

- **Case**

$$\frac{\Gamma_0 \longrightarrow \Omega_0 \qquad [\Omega_0]A = [\Omega_0]A'}{\Gamma_0, x : A \longrightarrow \Omega_0, x : A'}\ \longrightarrow\mathsf{Var}$$

$$\begin{array}{rl} \Gamma_0 \longrightarrow \Omega_0 & \text{Subderivation} \\ [\Omega_0]\Gamma_0 = [\Omega_0]\Omega_0 & \text{By i.h.} \\ [\Omega_0]A = [\Omega_0]A' & \text{Subderivation} \\ [\Omega_0]\Gamma_0, x : [\Omega_0]A = [\Omega_0]\Omega_0, x : [\Omega_0]A' & \text{By congruence of equality} \\ [\Omega_0, x : A'](\Gamma_0, x : A) = [\Omega_0, x : A'](\Omega_0, x : A') & \text{By definition of substitution} \end{array}$$

- **Case** $\dfrac{\Gamma_0 \longrightarrow \Omega_0}{\Gamma_0, \alpha : \kappa \longrightarrow \Omega_0, \alpha : \kappa} \longrightarrow\mathsf{Uvar}$

  Similar to $\longrightarrow$Var.

- **Case** $\dfrac{\Gamma_0 \longrightarrow \Omega_0}{\Gamma_0, \hat{\alpha} : \kappa \longrightarrow \Omega_0, \hat{\alpha} : \kappa} \longrightarrow\mathsf{Unsolved}$

  Similar to $\longrightarrow$Var.

- **Case** $\dfrac{\Gamma_0 \longrightarrow \Omega_0 \qquad [\Omega_0]t = [\Omega_0]t'}{\Gamma_0, \hat{\alpha} : \kappa = t \longrightarrow \Omega_0, \hat{\alpha} : \kappa = t'} \longrightarrow\mathsf{Solved}$

  Similar to $\longrightarrow$Var.

- **Case** $\dfrac{\Gamma_0 \longrightarrow \Omega_0}{\Gamma_0, \blacktriangleright_{\hat{\alpha}} \longrightarrow \Omega_0, \blacktriangleright_{\hat{\alpha}}} \longrightarrow\mathsf{Marker}$

  Similar to $\longrightarrow$Var.

- **Case** $\dfrac{\Gamma_0 \longrightarrow \Omega_0}{\Gamma_0, \widehat{\beta} : \kappa' \longrightarrow \Omega_0, \widehat{\beta} : \kappa' = t} \longrightarrow\mathsf{Solve}$

  Similar to $\longrightarrow$Var.

- **Case** $\dfrac{\Gamma_0 \longrightarrow \Omega_0 \qquad [\Omega_0]t' = [\Omega_0]t}{\Gamma_0, \alpha = t' \longrightarrow \Omega_0, \alpha = t} \longrightarrow\mathsf{Eqn}$

$\Gamma_0 \longrightarrow \Omega_0$	Subderivation
$[\Omega_0]t' = [\Omega_0]t$	Subderivation
$[\Omega_0]\Gamma_0 = [\Omega_0]\Omega_0$	By i.h.
$[[\Omega_0]t/\alpha]([\Omega_0]\Gamma_0) = [[\Omega_0]t/\alpha]([\Omega_0]\Omega_0)$	By congruence of equality
$[\Omega_0, \alpha = t](\Gamma_0, \alpha = t') = [\Omega_0, \alpha = t](\Omega_0, \alpha = t)$	By definition of context substitution

- **Case** $\dfrac{\Gamma \longrightarrow \Omega_0}{\Gamma \longrightarrow \Omega_0, \hat{\alpha} : \kappa} \longrightarrow\mathsf{Add}$

$\Gamma \longrightarrow \Omega_0$	Subderivation
$[\Omega_0]\Gamma = [\Omega_0]\Omega_0$	By i.h.
$[\Omega_0, \hat{\alpha} : \kappa]\Gamma = [\Omega_0, \hat{\alpha} : \kappa](\Omega_0, \hat{\alpha} : \kappa)$	By definition of context substitution

- **Case** $\dfrac{\Gamma \longrightarrow \Omega_0}{\Gamma \longrightarrow \Omega_0, \hat{\alpha} : \kappa = t} \longrightarrow\mathsf{AddSolved}$

  Similar to the $\longrightarrow$Add case. □

Lemma 55 (Completing Completeness)**.**

(i) *If* $\Omega \longrightarrow \Omega'$ *and* $\Omega \vdash t : \kappa$ *then* $[\Omega]t = [\Omega']t$.

(ii) *If* $\Omega \longrightarrow \Omega'$ *and* $\Omega \vdash A$ *type then* $[\Omega]A = [\Omega']A$.

(iii) *If* $\Omega \longrightarrow \Omega'$ *then* $[\Omega]\Omega = [\Omega']\Omega'$.

Proof.

- **Part (i):**

  By Lemma 29 (Substitution Monotonicity) (i), $[\Omega']t = [\Omega'][\Omega]t$.

  Now we need to show $[\Omega'][\Omega]t = [\Omega]t$. Considered as a substitution, Ω' is the identity everywhere except existential variables $\hat{\alpha}$ and universal variables α. First, since Ω is complete, $[\Omega]t$ has no free existentials. Second, universal variables free in $[\Omega]t$ have no equations in Ω (if they had, their occurrences would have been replaced). But if Ω has no equation for α, it follows from $\Omega \longrightarrow \Omega'$ and the definition of context extension in Figure **??** that Ω' also lacks an equation, so applying Ω' also leaves α alone.

  Transitivity of equality gives $[\Omega']t = [\Omega]t$.

- **Part (ii):** Similar to part (i), using Lemma 29 (Substitution Monotonicity) (iii) instead of (i).

- **Part (iii):** By induction on the given derivation of $\Omega \longrightarrow \Omega'$.

  Only cases $\longrightarrow$Id, $\longrightarrow$Var, $\longrightarrow$Uvar, $\longrightarrow$Eqn, $\longrightarrow$Solved, $\longrightarrow$AddSolved and $\longrightarrow$Marker are possible. In all of these cases, we use the i.h. and the definition of context application; in cases $\longrightarrow$Var, $\longrightarrow$Eqn and $\longrightarrow$Solved, we also use the equality in the premise of the respective rule. □

Lemma 56 (Confluence of Completeness)**.**
If $\Delta_1 \longrightarrow \Omega$ *and* $\Delta_2 \longrightarrow \Omega$ *then* $[\Omega]\Delta_1 = [\Omega]\Delta_2$.

Proof.

$\Delta_1 \longrightarrow \Omega$	Given
$[\Omega]\Delta_1 = [\Omega]\Omega$	By Lemma 54 (Completing Stability)
$\Delta_2 \longrightarrow \Omega$	Given
$[\Omega]\Delta_2 = [\Omega]\Omega$	By Lemma 54 (Completing Stability)
$[\Omega]\Delta_1 = [\Omega]\Delta_2$	By transitivity of equality

□

Lemma 57 (Multiple Confluence)**.**
If $\Delta \longrightarrow \Omega$ *and* $\Omega \longrightarrow \Omega'$ *and* $\Delta' \longrightarrow \Omega'$ *then* $[\Omega]\Delta = [\Omega']\Delta'$.

Proof.

$\Delta \longrightarrow \Omega$	Given
$[\Omega]\Delta = [\Omega]\Omega$	By Lemma 54 (Completing Stability)
$\Omega \longrightarrow \Omega'$	Given
$[\Omega]\Omega = [\Omega']\Omega'$	By Lemma 55 (Completing Completeness) (iii)
$= [\Omega']\Delta'$	By Lemma 54 (Completing Stability) ($\Delta' \longrightarrow \Omega'$ given)

□

Lemma 59 (Canonical Completion)**.**
If $\Gamma \longrightarrow \Omega$
then there exists Ω_{canon} *such that* $\Gamma \longrightarrow \Omega_{canon}$ *and* $\Omega_{canon} \longrightarrow \Omega$ *and* $\mathsf{dom}(\Omega_{canon}) = \mathsf{dom}(\Gamma)$ *and, for all* $\hat{\alpha} : \kappa = \tau$ *and* $\alpha = \tau$ *in* Ω_{canon}*, we have* $\mathsf{FEV}(\tau) = \emptyset$.

Proof. By induction on Ω. In Ω_{canon}, make all solutions (for evars and uvars) canonical by applying Ω to them, dropping declarations of existential variables that aren't in $\mathsf{dom}(\Gamma)$. □

Lemma 60 (Split Solutions)**.**
If $\Delta \longrightarrow \Omega$ *and* $\hat{\alpha} \in \mathsf{unsolved}(\Delta)$
then there exists $\Omega_1 = \Omega_1'[\hat{\alpha} : \kappa = t_1]$ *such that* $\Omega_1 \longrightarrow \Omega$ *and* $\Omega_2 = \Omega_1'[\hat{\alpha} : \kappa = t_2]$ *where* $\Delta \longrightarrow \Omega_2$ *and* $t_2 \neq t_1$ *and* Ω_2 *is canonical.*

Proof. Use Lemma 59 (Canonical Completion) to get Ω_{canon} such that $\Delta \longrightarrow \Omega_{canon}$ and $\Omega_{canon} \longrightarrow \Omega$, where for all solutions t in Ω_{canon} we have $\mathsf{FEV}(t) = \emptyset$.

We have $\Omega_{canon} = \Omega_1'[\hat{\alpha} : \kappa = t_1]$, where $\mathsf{FEV}(t_1) = \emptyset$. Therefore ☞ $\Omega_1'[\hat{\alpha} : \kappa = t_1] \longrightarrow \Omega$.

Now choose t_2 as follows:

- If $\kappa = \star$, let $t_2 = t_1 \to t_1$.
- If $\kappa = \mathbb{N}$, let $t_2 = \mathsf{succ}(t_1)$.

Thus, ☞ $t_2 \neq t_1$. Let $\Omega_2 = \Omega_1'[\hat{\alpha} : \kappa = t_2]$.

☞ $\Delta \longrightarrow \Omega_2$ By Lemma 31 (Split Extension) □

E′ Internal Properties of the Declarative System

Lemma 61 (Interpolating With and Exists)**.**

(1) If $\mathcal{D} :: \Psi \vdash \Pi :: \vec{A} \Leftarrow C\ p$ *and* $\Psi \vdash P_0$ *true*
then $\mathcal{D}' :: \Psi \vdash \Pi :: \vec{A} \Leftarrow C \wedge P_0\ p$.

(2) If $\mathcal{D} :: \Psi \vdash \Pi :: \vec{A} \Leftarrow [\tau/\alpha]C_0\ p$ *and* $\Psi \vdash \tau : \kappa$
then $\mathcal{D}' :: \Psi \vdash \Pi :: \vec{A} \Leftarrow (\exists \alpha : \kappa.\ C_0)\ p$.

In both cases, the height of $\mathcal{D}'$ *is one greater than the height of* $\mathcal{D}$.
Moreover, similar properties hold for the eliminating judgment $\Psi\ /\ P \vdash \Pi :: \vec{A} \Leftarrow C\ p$.

Proof. By induction on the given match derivation.
In the DeclMatchBase case, for part (1), apply rule $\wedge$I. For part (2), apply rule $\exists$I.
In the DeclMatchNeg case, part (1), use Lemma 2 (Declarative Weakening) (iii). In part (2), use Lemma 2 (Declarative Weakening) (i). □

Lemma 62 (Case Invertibility)**.**
If $\Psi \vdash \mathsf{case}(e_0, \Pi) \Leftarrow C\ p$
then $\Psi \vdash e_0 \Rightarrow A\ !$ *and* $\Psi \vdash \Pi :: A \Leftarrow C\ p$ *and* $\Psi \vdash \Pi$ *covers* A
where the height of each resulting derivation is strictly less than the height of the given derivation.

Proof. By induction on the given derivation.

- **Case** $\dfrac{\Psi \vdash \mathsf{case}(e_0, \Pi) \Rightarrow A\ q \qquad \mathrm{pol}(B) \vdash \Psi \leq^* A B}{\Psi \vdash \mathsf{case}(e_0, \Pi) \Leftarrow B\ p}\ \mathsf{DeclSub}$

  Impossible, because $\Psi \vdash \mathsf{case}(e_0, \Pi) \Rightarrow A\ q$ is not derivable.

- **Cases** Decl$\forall$I, Decl$\supset$I: Impossible: these rules have a value restriction, but a case expression is not a value.

- **Case** $\dfrac{\Psi \vdash P\ \mathit{true} \qquad \Psi \vdash \mathsf{case}(e_0, \Pi) \Leftarrow C_0\ p}{\Psi \vdash \mathsf{case}(e_0, \Pi) \Leftarrow C_0 \wedge P\ p}\ \mathsf{Decl}\wedge\mathsf{I}$

☞	$< n-1\ \ \Psi \vdash e_0 \Rightarrow A\ !$	By i.h.
	$< n-1\ \ \Psi \vdash \Pi :: A \Leftarrow C_0\ p$	$''$
☞	$< n-1\ \ \Psi \vdash \Pi$ *covers* A	$''$
	$\leq n-1\ \ \Psi \vdash P$ *true*	Subderivation
	☞ $< n\ \ \Psi \vdash \Pi :: A \Leftarrow C_0 \wedge P\ p$	By Lemma 61 (Interpolating With and Exists) (1)

- **Cases** Decl1I, Decl$\rightarrow$I, DeclRec, Decl$+$I$_k$, Decl$\times$I, DeclNil, DeclCons: Impossible, because in these rules e cannot have the form $\mathsf{case}(e_0, \Pi)$.

- **Case** $\dfrac{\Psi \vdash \mathsf{case}(e_0, \Pi) \Rightarrow A\ ! \qquad \Psi \vdash \Pi :: A \Leftarrow C\ p \qquad \Psi \vdash \Pi\ \mathit{covers}\ A}{\Psi \vdash \mathsf{case}(e_0, \Pi) \Leftarrow C\ p}\ \mathsf{DeclCase}$

  Immediate. □

F′ Miscellaneous Properties of the Algorithmic System

Lemma 63 (Well-Formed Outputs of Typing)**.**

(Spines) *If* $\Gamma \vdash s : A\ q \gg C\ p \dashv \Delta$ *or* $\Gamma \vdash s : A\ q \gg C\ \lceil p \rceil \dashv \Delta$
and $\Gamma \vdash A\ q$ *type*
then $\Delta \vdash C\ p$ *type*.

(Synthesis) *If* $\Gamma \vdash e \Rightarrow A\ p \dashv \Delta$
then $A \vdash p$ *type*.

Proof. By induction on the given derivation.

- **Case** Anno: Use Lemma 51 (Typing Extension) and Lemma 41 (Extension Weakening for Principal Typing).
- **Case** ∀Spine: We have $\Gamma \vdash (\forall \alpha : \kappa.\, A_0)\ \mathsf{q}\ \textit{type}$.
  By inversion, $\Gamma, \alpha : \kappa \vdash A_0\ \mathsf{q}\ \textit{type}$.
  By properties of substitution, $\Gamma, \hat{\alpha} : \kappa \vdash [\hat{\alpha}/\alpha]A_0\ \mathsf{q}\ \textit{type}$.
  Now apply the i.h.
- **Case** ⊃Spine: Use Lemma 42 (Inversion of Principal Typing) (2), Lemma 47 (Checkprop Extension), and Lemma 41 (Extension Weakening for Principal Typing).
- **Case** SpineRecover:
  By i.h., $\Delta \vdash C \not\parallel \textit{type}$.
  We have as premise $\mathsf{FEV}(C) = \emptyset$.
  Therefore $\Delta \vdash C\ !\ \textit{type}$.
- **Case** SpinePass: By i.h.
- **Case** EmptySpine: Immediate.
- **Case** →Spine: Use Lemma 42 (Inversion of Principal Typing) (1), Lemma 51 (Typing Extension), and Lemma 41 (Extension Weakening for Principal Typing).
- **Case** $\hat{\alpha}$Spine: Show that $\hat{\alpha}_1 \to \hat{\alpha}_2$ is well-formed, then use the i.h. □

G′ Decidability of Instantiation

Lemma 64 (Left Unsolvedness Preservation)**.**
If $\underbrace{\Gamma_0, \hat{\alpha}, \Gamma_1}_{\Gamma} \vdash \hat{\alpha} := A : \kappa \dashv \Delta$ *and* $\hat{\beta} \in \mathsf{unsolved}(\Gamma_0)$ *then* $\hat{\beta} \in \mathsf{unsolved}(\Delta)$.

Proof. By induction on the given derivation.

- **Case**
$$\frac{\Gamma_0 \vdash \tau : \kappa}{\underbrace{\Gamma_0, \hat{\alpha} : \kappa, \Gamma_1}_{\Gamma} \vdash \hat{\alpha} := \tau : \kappa \dashv \underbrace{\Gamma_0, \hat{\alpha} : \kappa = \tau, \Gamma_1}_{\Delta}}\ \mathsf{InstSolve}$$
  Immediate, since to the left of $\hat{\alpha}$, the contexts Δ and Γ are the same.
- **Case**
$$\frac{\hat{\beta} \in \mathsf{unsolved}(\Gamma'[\hat{\alpha} : \kappa][\hat{\beta} : \kappa])}{\underbrace{\Gamma'[\hat{\alpha} : \kappa][\hat{\beta} : \kappa]}_{\Gamma} \vdash \hat{\alpha} := \hat{\beta} : \kappa \dashv \underbrace{\Gamma'[\hat{\alpha} : \kappa][\hat{\beta} : \kappa = \hat{\alpha}]}_{\Delta}}\ \mathsf{InstReach}$$
  Immediate, since to the left of $\hat{\alpha}$, the contexts Δ and Γ are the same.
- **Case**
$$\frac{\Gamma_0, \hat{\alpha}_2 : \star, \hat{\alpha}_1 : \star, \hat{\alpha} : \star = \hat{\alpha}_1 \oplus \hat{\alpha}_2, \Gamma_1 \vdash \hat{\alpha}_1 := \tau_1 : \star \dashv \Theta \qquad \Theta \vdash \hat{\alpha}_2 := [\Theta]\tau_2 : \star \dashv \Delta}{\Gamma_0, \hat{\alpha} : \star, \Gamma_1 \vdash \hat{\alpha} := \tau_1 \oplus \tau_2 : \star \dashv \Delta}\ \mathsf{InstBin}$$
  We have $\hat{\beta} \in \mathsf{unsolved}(\Gamma_0)$. Therefore $\hat{\beta} \in \mathsf{unsolved}(\Gamma_0, \hat{\alpha}_2 : \star)$.
  Clearly, $\hat{\alpha}_2 \in \mathsf{unsolved}(\Gamma_0, \hat{\alpha}_2 : \star)$.
  We have two subderivations:
$$\Gamma_0, \hat{\alpha}_2 : \star, \hat{\alpha}_1 : \star, \hat{\alpha} : \star = \hat{\alpha}_1 \oplus \hat{\alpha}_2, \Gamma_1 \vdash \hat{\alpha}_1 := A_1 : \star \dashv \Theta \tag{1}$$
$$\Theta \vdash \hat{\alpha}_2 := [\Theta]A_2 : \star \dashv \Delta \tag{2}$$
  By induction on (1), $\hat{\beta} \in \mathsf{unsolved}(\Theta)$.
  Also by induction on (1), with $\hat{\alpha}_2$ playing the role of $\hat{\beta}$, we get $\hat{\alpha}_2 \in \mathsf{unsolved}(\Theta)$.
  Since $\hat{\beta} \in \Gamma_0$, it is declared to the left of $\hat{\alpha}_2$ in $\Gamma_0, \hat{\alpha}_2 : \star, \hat{\alpha}_1 : \star, \hat{\alpha} = \hat{\alpha}_1 \oplus \hat{\alpha}_2, \Gamma_1$.

Hence by Lemma 20 (Declaration Order Preservation), $\hat{\beta}$ is declared to the left of $\hat{\alpha}_2$ in Θ. That is, $\Theta = (\Theta_0, \hat{\alpha}_2 : \star, \Theta_1)$, where $\hat{\beta} \in \mathsf{unsolved}(\Theta_0)$.
By induction on (2), $\hat{\beta} \in \mathsf{unsolved}(\Delta)$.

- **Case**

$$\frac{}{\underbrace{\Gamma'[\hat{\alpha} : \mathbb{N}]}_{\Gamma} \vdash \hat{\alpha} := \mathsf{zero} : \mathbb{N} \dashv \underbrace{\Gamma'[\hat{\alpha} : \mathbb{N} = \mathsf{zero}]}_{\Delta}}\ \mathsf{InstZero}$$

Immediate, since to the left of $\hat{\alpha}$, the contexts Δ and Γ are the same.

- **Case** $\dfrac{\Gamma[\hat{\alpha}_1 : \mathbb{N}, \hat{\alpha} : \mathbb{N} = \mathsf{succ}(\hat{\alpha}_1)] \vdash \hat{\alpha}_1 := t_1 : \mathbb{N} \dashv \Delta}{\Gamma[\hat{\alpha} : \mathbb{N}] \vdash \hat{\alpha} := \mathsf{succ}(t_1) : \mathbb{N} \dashv \Delta}\ \mathsf{InstSucc}$

We have $\hat{\beta} \in \mathsf{unsolved}(\Gamma_0)$. Therefore $\hat{\beta} \in \mathsf{unsolved}(\Gamma_0, \hat{\alpha}_1 : \mathbb{N})$. By i.h., $\hat{\beta} \in \mathsf{unsolved}(\Delta)$. □

Lemma 65 (Left Free Variable Preservation)**.** *If* $\overbrace{\Gamma_0, \hat{\alpha} : \kappa, \Gamma_1}^{\Gamma} \vdash \hat{\alpha} := t : \kappa \dashv \Delta$ *and* $\Gamma \vdash s : \kappa'$ *and* $\hat{\alpha} \notin \mathrm{FV}([\Gamma]s)$ *and* $\hat{\beta} \in \mathsf{unsolved}(\Gamma_0)$ *and* $\hat{\beta} \notin \mathrm{FV}([\Gamma]s)$*, then* $\hat{\beta} \notin \mathrm{FV}([\Delta]s)$.

Proof. By induction on the given instantiation derivation.

- **Case**

$$\frac{\Gamma_0 \vdash \tau : \kappa}{\Gamma_0, \hat{\alpha} : \kappa, \Gamma_1 \vdash \hat{\alpha} := \tau : \kappa \dashv \underbrace{\Gamma_0, \hat{\alpha} : \kappa = \tau, \Gamma_1}_{\Delta}}\ \mathsf{InstSolve}$$

We have $\hat{\alpha} \notin \mathrm{FV}([\Gamma]\sigma)$. Since Δ differs from Γ only in $\hat{\alpha}$, it must be the case that $[\Gamma]\sigma = [\Delta]\sigma$. It is given that $\hat{\beta} \notin \mathrm{FV}([\Gamma]\sigma)$, so $\hat{\beta} \notin \mathrm{FV}([\Delta]\sigma)$.

- **Case**

$$\frac{\hat{\gamma} \in \mathsf{unsolved}(\Gamma[\hat{\alpha} : \kappa][\hat{\gamma} : \kappa])}{\Gamma[\hat{\alpha} : \kappa][\hat{\gamma} : \kappa] \vdash \hat{\alpha} := \hat{\gamma} : \kappa \dashv \underbrace{\Gamma[\hat{\alpha} : \kappa][\hat{\gamma} : \kappa = \hat{\alpha}]}_{\Delta}}\ \mathsf{InstReach}$$

Since Δ differs from Γ only in solving $\hat{\gamma}$ to $\hat{\alpha}$, applying Δ to a type will not introduce a $\hat{\beta}$. We have $\hat{\beta} \notin \mathrm{FV}([\Gamma]\sigma)$, so $\hat{\beta} \notin \mathrm{FV}([\Delta]\sigma)$.

- **Case**

$$\frac{\overbrace{\Gamma[\hat{\alpha}_2 : \star, \hat{\alpha}_1 : \star, \hat{\alpha} : \star = \hat{\alpha}_1 \oplus \hat{\alpha}_2]}^{\Gamma'} \vdash \hat{\alpha}_1 := \tau_1 : \star \dashv \Theta \qquad \Theta \vdash \hat{\alpha}_2 := [\Theta]\tau_2 : \star \dashv \Delta}{\Gamma[\hat{\alpha} : \star] \vdash \hat{\alpha} := \tau_1 \oplus \tau_2 : \star \dashv \Delta}\ \mathsf{InstBin}$$

We have $\Gamma \vdash \sigma$ *type* and $\hat{\alpha} \notin \mathrm{FV}([\Gamma]\sigma)$ and $\hat{\beta} \notin \mathrm{FV}([\Gamma]\sigma)$.
By weakening, we get $\Gamma' \vdash \sigma : \kappa'$; since $\hat{\alpha} \notin \mathrm{FV}([\Gamma]\sigma)$ and Γ' only adds a solution for $\hat{\alpha}$, it follows that $[\Gamma']\sigma = [\Gamma]\sigma$.
Therefore $\hat{\alpha}_1 \notin \mathrm{FV}([\Gamma']\sigma)$ and $\hat{\alpha}_2 \notin \mathrm{FV}([\Gamma']\sigma)$ and $\hat{\beta} \notin \mathrm{FV}([\Gamma']\sigma)$.
Since we have $\hat{\beta} \in \Gamma_0$, we also have $\hat{\beta} \in (\Gamma_0, \hat{\alpha}_2 : \star)$.
By induction on the first premise, $\hat{\beta} \notin \mathrm{FV}([\Theta]\sigma)$.
Also by induction on the first premise, with $\hat{\alpha}_2$ playing the role of $\hat{\beta}$, we have $\hat{\alpha}_2 \notin \mathrm{FV}([\Theta]\sigma)$.
Note that $\hat{\alpha}_2 \in \mathsf{unsolved}(\Gamma_0, \hat{\alpha}_2 : \star)$.
By Lemma 64 (Left Unsolvedness Preservation), $\hat{\alpha}_2 \in \mathsf{unsolved}(\Theta)$.
Therefore Θ has the form $(\Theta_0, \hat{\alpha}_2 : \star, \Theta_1)$.
Since $\hat{\beta} \neq \hat{\alpha}_2$, we know that $\hat{\beta}$ is declared to the left of $\hat{\alpha}_2$ in $(\Gamma_0, \hat{\alpha}_2 : \star)$, so by Lemma 20 (Declaration Order Preservation), $\hat{\beta}$ is declared to the left of $\hat{\alpha}_2$ in Θ. Hence $\hat{\beta} \in \Theta_0$.
Furthermore, by Lemma 43 (Instantiation Extension), we have $\Gamma' \longrightarrow \Theta$.
Then by Lemma 36 (Extension Weakening (Sorts)), we have $\Delta \vdash \sigma : \kappa'$.
Using induction on the second premise, $\hat{\beta} \notin \mathrm{FV}([\Delta]\sigma)$.

- **Case**

$$\frac{}{\underbrace{\Gamma'[\hat{\alpha} : \mathbb{N}]}_{\Gamma} \vdash \hat{\alpha} := \mathsf{zero} : \mathbb{N} \dashv \underbrace{\Gamma'[\hat{\alpha} : \mathbb{N} = \mathsf{zero}]}_{\Delta}}\ \mathsf{InstZero}$$

We have $\hat{\alpha} \notin \mathrm{FV}([\Gamma]\sigma)$. Since Δ differs from Γ only in $\hat{\alpha}$, it must be the case that $[\Gamma]\sigma = [\Delta]\sigma$. It is given that $\hat{\beta} \notin \mathrm{FV}([\Gamma]\sigma)$, so $\hat{\beta} \notin \mathrm{FV}([\Delta]\sigma)$.

- **Case**

$$\frac{\overbrace{\Gamma'[\hat{\alpha}_1 : \mathbb{N}, \hat{\alpha} : \mathbb{N} = \mathsf{succ}(\hat{\alpha}_1)]}^{\Theta} \vdash \hat{\alpha}_1 := \mathsf{t}_1 : \mathbb{N} \dashv \Delta}{\underbrace{\Gamma'[\hat{\alpha} : \mathbb{N}]}_{\Gamma} \vdash \hat{\alpha} := \mathsf{succ}(\mathsf{t}_1) : \mathbb{N} \dashv \Delta}\ \mathsf{InstSucc}$$

$\Gamma \vdash \sigma : \kappa'$	Given
$\Theta \vdash \sigma : \kappa'$	By weakening
$\hat{\alpha} \notin \mathrm{FV}([\Gamma]\sigma)$	Given
$\hat{\alpha} \notin \mathrm{FV}([\Theta]\sigma)$	$\hat{\alpha} \notin \mathrm{FV}([\Gamma]\sigma)$ and Θ only solves $\hat{\alpha}$
$\Theta = (\Gamma_0, \hat{\alpha}_1 : \mathbb{N}, \hat{\alpha} : \mathbb{N} = \mathsf{succ}(\hat{\alpha}_1), \Gamma_1)$	Given
$\hat{\beta} \notin \mathsf{unsolved}(\Gamma_0)$	Given
$\hat{\beta} \notin \mathsf{unsolved}(\Gamma_0, \hat{\alpha}_1 : \mathbb{N})$	$\hat{\alpha}_1$ fresh
$\hat{\beta} \notin \mathrm{FV}([\Gamma]\sigma)$	Given
$\hat{\beta} \notin \mathrm{FV}([\Theta]\sigma)$	$\hat{\alpha}_1$ fresh
☞ $\hat{\beta} \notin \mathrm{FV}([\Delta]\sigma)$	By i.h.

□

Lemma 66 (Instantiation Size Preservation)**.** *If* $\overbrace{\Gamma_0, \hat{\alpha}, \Gamma_1}^{\Gamma} \vdash \hat{\alpha} := \tau : \kappa \dashv \Delta$ *and* $\Gamma \vdash s : \kappa'$ *and* $\hat{\alpha} \notin \mathrm{FV}([\Gamma]s)$*, then* $|[\Gamma]s| = |[\Delta]s|$*, where* $|C|$ *is the plain size of the term* C*.*

Proof. By induction on the given derivation.

- **Case**

$$\frac{\Gamma_0 \vdash \tau : \kappa}{\underbrace{\Gamma_0, \hat{\alpha} : \kappa, \Gamma_1}_{\Gamma} \vdash \hat{\alpha} := \tau : \kappa \dashv \underbrace{\Gamma_0, \hat{\alpha} : \kappa = \tau, \Gamma_1}_{\Delta}}\ \mathsf{InstSolve}$$

Since Δ differs from Γ only in solving $\hat{\alpha}$, and we know $\hat{\alpha} \notin \mathrm{FV}([\Gamma]\sigma)$, we have $[\Delta]\sigma = [\Gamma]\sigma$; therefore $|[\Delta]\sigma = [\Gamma]\sigma|$.

- **Case**

$$\frac{}{\underbrace{\Gamma'[\hat{\alpha} : \mathbb{N}]}_{\Gamma} \vdash \hat{\alpha} := \mathsf{zero} : \mathbb{N} \dashv \underbrace{\Gamma'[\hat{\alpha} : \mathbb{N} = \mathsf{zero}]}_{\Delta}}\ \mathsf{InstZero}$$

Similar to the InstSolve case.

- **Case**

$$\frac{\hat{\beta} \in \mathsf{unsolved}(\Gamma'[\hat{\alpha} : \kappa][\hat{\beta} : \kappa])}{\underbrace{\Gamma'[\hat{\alpha} : \kappa][\hat{\beta} : \kappa]}_{\Gamma} \vdash \hat{\alpha} := \hat{\beta} : \kappa \dashv \underbrace{\Gamma'[\hat{\alpha} : \kappa][\hat{\beta} : \kappa = \hat{\alpha}]}_{\Delta}}\ \mathsf{InstReach}$$

Here, Δ differs from Γ only in solving $\hat{\beta}$ to $\hat{\alpha}$. However, $\hat{\alpha}$ has the same size as $\hat{\beta}$, so even if $\hat{\beta} \in \mathrm{FV}([\Gamma]\sigma)$, we have $|[\Delta]\sigma = [\Gamma]\sigma|$.

- **Case**

$$\frac{\overbrace{\Gamma[\hat{\alpha}_2 : \star, \hat{\alpha}_1 : \star, \hat{\alpha} : \star = \hat{\alpha}_1 \oplus \hat{\alpha}_2]}^{\Gamma'} \vdash \hat{\alpha}_1 := \tau_1 : \star \dashv \Theta \qquad \Theta \vdash \hat{\alpha}_2 := [\Theta]\tau_2 : \star \dashv \Delta}{\Gamma[\hat{\alpha} : \star] \vdash \hat{\alpha} := \tau_1 \oplus \tau_2 : \star \dashv \Delta}\ \mathsf{InstBin}$$

We have $\Gamma \vdash \sigma : \kappa'$ and $\hat{\alpha} \notin \mathrm{FV}([\Gamma]\sigma)$.
Since $\hat{\alpha}_1, \hat{\alpha}_2 \notin \mathsf{dom}(\Gamma)$, we have $\hat{\alpha}, \hat{\alpha}_1, \hat{\alpha}_2 \notin \mathrm{FV}([\Gamma]\sigma)$.
By Lemma 23 (Deep Evar Introduction), $\Gamma[\hat{\alpha} : \star] \longrightarrow \Gamma'$.
By Lemma 36 (Extension Weakening (Sorts)), $\Gamma' \vdash \sigma : \kappa'$.
Since $\hat{\alpha} \notin \mathrm{FV}(\sigma)$, it follows that $[\Gamma']\sigma = [\Gamma]\sigma$, and so $|[\Gamma']\sigma| = |[\Gamma]\sigma|$.

By induction on the first premise, $|[\Gamma']\sigma| = |[\Theta]\sigma|$.
By Lemma 20 (Declaration Order Preservation), since $\hat{\alpha}_2$ is declared to the left of $\hat{\alpha}_1$ in Γ', we have that $\hat{\alpha}_2$ is declared to the left of $\hat{\alpha}_1$ in Θ.
By Lemma 64 (Left Unsolvedness Preservation), since $\hat{\alpha}_2 \in \mathsf{unsolved}(\Gamma')$, it is unsolved in Θ: that is, $\Theta = (\Theta_0, \hat{\alpha}_2 : \star, \Theta_1)$.
By Lemma 43 (Instantiation Extension), we have $\Gamma' \longrightarrow \Theta$.
By Lemma 36 (Extension Weakening (Sorts)), $\Theta \vdash \sigma : \kappa'$.
Since $\hat{\alpha}_2 \notin \mathrm{FV}([\Gamma']\sigma)$, Lemma 65 (Left Free Variable Preservation) gives $\hat{\alpha}_2 \notin \mathrm{FV}([\Theta]\sigma)$.
By induction on the second premise, $|[\Theta]\sigma| = |[\Delta]\sigma|$, and by transitivity of equality, $|[\Gamma]\sigma| = |[\Delta]\sigma|$.

- **Case**
$$\frac{\overbrace{\Gamma[\hat{\alpha}_1 : \mathbb{N}, \hat{\alpha} : \mathbb{N} = \mathsf{succ}(\hat{\alpha}_1)]}^{\Gamma'} \vdash \hat{\alpha}_1 := \mathrm{t}_1 : \mathbb{N} \dashv \Delta}{\Gamma[\hat{\alpha} : \mathbb{N}] \vdash \hat{\alpha} := \mathsf{succ}(\mathrm{t}_1) : \mathbb{N} \dashv \Delta}\ \mathsf{InstSucc}$$

$\Gamma[\hat{\alpha} : \star] \vdash \sigma : \kappa'$	Given
$\hat{\alpha} \notin [\Gamma[\hat{\alpha} : \star]]\sigma$	Given
$\Gamma[\hat{\alpha} : \star] \longrightarrow \Gamma'$	By Lemma 23 (Deep Evar Introduction)
$\Gamma' \vdash \sigma : \kappa'$	By Lemma 36 (Extension Weakening (Sorts))
$[\Gamma']\sigma = [\Gamma[\hat{\alpha} : \star]]\sigma$	Since $\hat{\alpha} \notin \mathrm{FV}([\Gamma[\hat{\alpha} : \star]]\sigma)$
$\lvert[\Gamma']\sigma\rvert = \lvert[\Gamma[\hat{\alpha} : \star]]\sigma\rvert$	By congruence of equality
$\hat{\alpha}_1 \notin [\Gamma']\sigma$	Since $[\Gamma']\sigma = [\Gamma[\hat{\alpha} : \star]]\sigma$, and $\hat{\alpha}_1 \notin \mathsf{dom}(\Gamma[\hat{\alpha} : \star])$
$\lvert[\Gamma']\sigma\rvert = \lvert[\Theta]\sigma\rvert$	By i.h.
$\lvert[\Gamma[\hat{\alpha} : \star]]\sigma\rvert = \lvert[\Theta]\sigma\rvert$	By transitivity of equality

□

Lemma 67 (Decidability of Instantiation)**.** *If* $\Gamma = \Gamma_0[\hat{\alpha} : \kappa']$ *and* $\Gamma \vdash \mathrm{t} : \kappa$ *such that* $[\Gamma]\mathrm{t} = \mathrm{t}$ *and* $\hat{\alpha} \notin \mathrm{FV}(\mathrm{t})$*, then:*

(1) Either there exists Δ *such that* $\Gamma_0[\hat{\alpha} : \kappa'] \vdash \hat{\alpha} := \mathrm{t} : \kappa \dashv \Delta$*, or not.*

Proof. By induction on the derivation of $\Gamma \vdash \mathrm{t} : \kappa$.

- **Case**
$$\frac{(\mathrm{u} : \kappa) \in \Gamma}{\Gamma_L, \hat{\alpha} : \kappa', \Gamma_R \vdash \mathrm{u} : \kappa}\ \mathsf{VarSort}$$

If $\kappa \neq \kappa'$, no rule matches and no derivation exists.
Otherwise:

  - If $(\mathrm{u} : \kappa) \in \Gamma_L$, we can apply rule InstSolve.
  - If u is some unsolved existential variable $\hat{\beta}$ and $(\hat{\beta} : \kappa) \in \Gamma_R$, then we can apply rule InstReach.
  - Otherwise, u is declared in Γ_R and is a universal variable; no rule matches and no derivation exists.

- **Case**
$$\frac{(\hat{\beta} : \kappa = \tau) \in \Gamma}{\Gamma \vdash \hat{\beta} : \kappa}\ \mathsf{SolvedVarSort}$$

By inversion, $(\hat{\beta} : \kappa = \tau) \in \Gamma$, but $[\Gamma]\hat{\beta} = \hat{\beta}$ is given, so this case is impossible.

- **Case** UnitSort:
If $\kappa' = \star$, then apply rule InstSolve. Otherwise, no rule matches and no derivation exists.

- **Case**
$$\frac{\Gamma \vdash \tau_1 : \star \qquad \Gamma \vdash \tau_2 : \star}{\underbrace{\Gamma_L, \hat{\alpha} : \kappa', \Gamma_R}_{\Gamma} \vdash \tau_1 \oplus \tau_2 : \star}\ \mathsf{BinSort}$$

If $\kappa' \neq \star$, then no rule matches and no derivation exists. Otherwise:
Given, $[\Gamma](\tau_1 \oplus \tau_2) = \tau_1 \oplus \tau_2$ and $\hat{\alpha} \notin \mathrm{FV}([\Gamma](\tau_1 \oplus \tau_2))$.
If $\Gamma_L \vdash \tau_1 \oplus \tau_2 : \star$, then we have a derivation by InstSolve.

If not, the only other rule whose conclusion matches $\tau_1 \oplus \tau_2$ is InstBin.
First, consider whether $\Gamma_L, \hat{\alpha}_2 : \star, \hat{\alpha}_1 : \star, \hat{\alpha} : \star = \hat{\alpha}_1 \oplus \hat{\alpha}_2, \Gamma_R \vdash \hat{\alpha}_1 := t : \star \dashv -$ is decidable.

By definition of substitution, $[\Gamma](\tau_1 \oplus \tau_2) = ([\Gamma]\tau_1) \oplus ([\Gamma]\tau_2)$. Since $[\Gamma](\tau_1 \oplus \tau_2) = \tau_1 \oplus \tau_2$, we have $[\Gamma]\tau_1 = \tau_1$ and $[\Gamma]\tau_2 = \tau_2$.
By weakening, $\Gamma_L, \hat{\alpha}_2 : \star, \hat{\alpha}_1 : \star, \hat{\alpha} : \star = \hat{\alpha}_1 \oplus \hat{\alpha}_2, \Gamma_R \vdash \tau_1 \oplus \tau_2 : \star$.
Since $\Gamma \vdash \tau_1 : \star$ and $\Gamma \vdash \tau_2 : \star$, we have $\hat{\alpha}_1, \hat{\alpha}_2 \notin \mathrm{FV}(\tau_1) \cup \mathrm{FV}(\tau_2)$.
Since $\hat{\alpha} \notin \mathrm{FV}(t) \supseteq \mathrm{FV}(\tau_1)$, it follows that $[\Gamma']\tau_1 = \tau_1$.
By i.h., either there exists Θ s.t. $\Gamma_L, \hat{\alpha}_2 : \star, \hat{\alpha}_1 : \star, \hat{\alpha} : \star = \hat{\alpha}_1 \oplus \hat{\alpha}_2, \Gamma_R \vdash \hat{\alpha}_1 := \tau_1 : \star \dashv \Theta$, or not.
If not, then no derivation by InstBin exists.
Otherwise, there exists such a Θ. By Lemma 64 (Left Unsolvedness Preservation), we have $\hat{\alpha}_2 \in \mathsf{unsolved}(\Theta)$.
By Lemma 65 (Left Free Variable Preservation), we know that $\hat{\alpha}_2 \notin \mathrm{FV}([\Theta]\tau_2)$.
Substitution is idempotent, so $[\Theta][\Theta]\tau_2 = [\Theta]\tau_2$.
By i.h., either there exists Δ such that $\Theta \vdash \hat{\alpha}_2 := [\Theta]\tau_2 : \kappa \dashv \Delta$, or not.
If not, no derivation by InstBin exists.
Otherwise, there exists such a Δ. By rule InstBin, we have $\Gamma \vdash \hat{\alpha} := t : \kappa \dashv \Delta$.

- **Case**

$$\frac{}{\Gamma \vdash \mathsf{zero} : \mathbb{N}}\ \mathsf{ZeroSort}$$

If $\kappa' \neq \mathbb{N}$, then no rule matches and no derivation exists. Otherwise, apply rule InstSolve.

- **Case**

$$\frac{\Gamma \vdash t_0 : \mathbb{N}}{\Gamma \vdash \mathsf{succ}(t_0) : \mathbb{N}}\ \mathsf{SuccSort}$$

If $\kappa' \neq \mathbb{N}$, then no rule matches and no derivation exists. Otherwise:
If $\Gamma_L \vdash \mathsf{succ}(t_0) : \mathbb{N}$, then we have a derivation by InstSolve.
If not, the only other rule whose conclusion matches $\mathsf{succ}(t_0)$ is InstSucc.
The remainder of this case is similar to the BinSort case, but shorter. □

H′ Separation

Lemma 68 (Transitivity of Separation)**.**
If $(\Gamma_L * \Gamma_R) \xrightarrow{*} (\Theta_L * \Theta_R)$ *and* $(\Theta_L * \Theta_R) \xrightarrow{*} (\Delta_L * \Delta_R)$
then $(\Gamma_L * \Gamma_R) \xrightarrow{*} (\Delta_L * \Delta_R)$.

Proof.

	$(\Gamma_L * \Gamma_R) \xrightarrow{*} (\Theta_L * \Theta_R)$	Given
	$(\Gamma_L, \Gamma_R) \longrightarrow (\Theta_L, \Theta_R)$	By Definition 5
	$\Gamma_L \subseteq \Theta_L$ and $\Gamma_R \subseteq \Theta_R$	″
	$(\Theta_L * \Theta_R) \xrightarrow{*} (\Delta_L * \Delta_R)$	Given
	$(\Theta_L, \Theta_R) \longrightarrow (\Delta_L, \Delta_R)$	By Definition 5
	$\Theta_L \subseteq \Delta_L$ and $\Theta_R \subseteq \Delta_R$	″
	$(\Gamma_L, \Gamma_R) \longrightarrow (\Delta_L, \Delta_R)$	By Lemma 33 (Extension Transitivity)
	$\Gamma_L \subseteq \Delta_L$ and $\Gamma_R \subseteq \Delta_R$	By transitivity of $\subseteq$
☞	$(\Gamma_L * \Gamma_R) \xrightarrow{*} (\Delta_L * \Delta_R)$	By Definition 5

□

Lemma 69 (Separation Truncation)**.**
If H *has the form* $\alpha : \kappa$ *or* $\blacktriangleright_{\hat{\alpha}}$ *or* $\blacktriangleright_P$ *or* $x : A\ p$
and $(\Gamma_L * (\Gamma_R, H)) \xrightarrow{*} (\Delta_L * \Delta_R)$
then $(\Gamma_L * \Gamma_R) \xrightarrow{*} (\Delta_L * \Delta_0)$ *where* $\Delta_R = (\Delta_0, H, \Theta)$.

Proof. By induction on Δ_R.

If $\Delta_R = (\ldots, H)$, we have $(\Gamma_L * \Gamma_R, H) \xrightarrow{*} (\Delta_L * (\Delta, H))$, and inversion on ⟶Uvar (if H is $(\alpha : \kappa)$, or the corresponding rule for other forms) gives the result (with $\Theta = \cdot$).

Otherwise, proceed into the subderivation of $(\Gamma_L, \Gamma_R, \alpha : \kappa) \longrightarrow (\Delta_L, \Delta_R)$, with $\Delta_R = (\Delta'_R, \Delta')$ where Δ' is a single declaration. Use the i.h. on Δ'_R, producing some Θ'. Finally, let $\Theta = (\Theta', \Delta')$. ☐

Lemma 70 (Separation for Auxiliary Judgments).

(i) *If* $\Gamma_L * \Gamma_R \vdash \sigma \stackrel{\circ}{=} \tau : \kappa \dashv \Delta$
and $\mathsf{FEV}(\sigma) \cup \mathsf{FEV}(\tau) \subseteq \mathsf{dom}(\Gamma_R)$
then $\Delta = (\Delta_L * \Delta_R)$ *and* $(\Gamma_L * \Gamma_R) \xrightarrow[*]{} (\Delta_L * \Delta_R)$.

(ii) *If* $\Gamma_L * \Gamma_R \vdash P$ *true* $\dashv \Delta$
and $\mathsf{FEV}(P) \subseteq \mathsf{dom}(\Gamma_R)$
then $\Delta = (\Delta_L * \Delta_R)$ *and* $(\Gamma_L * \Gamma_R) \xrightarrow[*]{} (\Delta_L * \Delta_R)$.

(iii) *If* $\Gamma_L * \Gamma_R \ / \ \sigma \stackrel{\circ}{=} \tau : \kappa \dashv \Delta$
and $\mathsf{FEV}(\sigma) \cup \mathsf{FEV}(\tau) = \emptyset$
then $\Delta = (\Delta_L * (\Delta_R, \Theta))$ *and* $(\Gamma_L * (\Gamma_R, \Theta)) \xrightarrow[*]{} (\Delta_L * \Delta_R)$.

(iv) *If* $\Gamma_L * \Gamma_R \ / \ P \dashv \Delta$
and $\mathsf{FEV}(P) = \emptyset$
then $\Delta = (\Delta_L * (\Delta_R, \Theta))$ *and* $(\Gamma_L * (\Gamma_R, \Theta)) \xrightarrow[*]{} (\Delta_L * \Delta_R)$.

(v) *If* $\Gamma_L * \Gamma_R \vdash \hat{\alpha} := \tau : \kappa \dashv \Delta$
and $(\mathsf{FEV}(\tau) \cup \{\hat{\alpha}\}) \subseteq \mathsf{dom}(\Gamma_R)$
then $\Delta = (\Delta_L * \Delta_R)$ *and* $(\Gamma_L * \Gamma_R) \xrightarrow[*]{} (\Delta_L * \Delta_R)$.

(vi) *If* $\Gamma_L * \Gamma_R \vdash P \equiv Q \dashv \Delta$
and $\mathsf{FEV}(P) \cup \mathsf{FEV}(Q) \subseteq \mathsf{dom}(\Gamma_R)$
then $\Delta = (\Delta_L * \Delta_R)$ *and* $(\Gamma_L * \Gamma_R) \xrightarrow[*]{} (\Delta_L * \Delta_R)$.

(vii) *If* $\Gamma_L * \Gamma_R \vdash A \equiv B \dashv \Delta$
and $\mathsf{FEV}(A) \cup \mathsf{FEV}(B) \subseteq \mathsf{dom}(\Gamma_R)$
then $\Delta = (\Delta_L * \Delta_R)$ *and* $(\Gamma_L * \Gamma_R) \xrightarrow[*]{} (\Delta_L * \Delta_R)$.

Proof. Part (i): By induction on the derivation of the given checkeq judgment. Cases CheckeqVar, CheckeqUnit and CheckeqZero are immediate ($\Delta_L = \Gamma_L$ and $\Delta_R = \Gamma_R$). For case CheckeqSucc, apply the i.h. For cases CheckeqInstL and CheckeqInstR, use the i.h. (v). For case CheckeqBin, use reasoning similar to that in the $\wedge$I case of Lemma 72 (Separation—Main) (transitivity of separation, and applying Θ in the second premise).

Part (ii), checkprop: Use the i.h. (i).

Part (iii), elimeq: Cases ElimeqUvarRefl, ElimeqUnit and CheckeqZero are immediate ($\Delta_L = \Gamma_L$ and $\Delta_R = \Gamma_R$). Cases ElimeqUvarL$\bot$, ElimeqUvarR$\bot$, ElimeqBinBot and ElimeqClash are impossible (we have Δ, not $\bot$). For case ElimeqSucc, apply the i.h. The case for ElimeqBin is similar to the case CheckeqBin in part (i). For cases ElimeqUvarL and ElimeqUvarR, $\Delta = (\Gamma_L, \Gamma_R, \alpha = \tau)$ which, since $\mathsf{FEV}(\tau) \subseteq \mathsf{dom}(\Gamma_R)$, ensures that $(\Gamma_L * (\Gamma_R, \alpha = \tau)) \xrightarrow[*]{} (\Delta_L * (\Delta_R, \alpha = \tau))$.

Part (iv), elimprop: Use the i.h. (iii).

Part (v), instjudg:

- **Case** InstSolve: Here, $\Gamma = (\Gamma_0, \hat{\alpha} : \kappa, \Gamma_1)$ and $\Delta = (\Gamma_0, \hat{\alpha} : \kappa = \tau, \Gamma_1)$. We have $\hat{\alpha} \in \mathsf{dom}(\Gamma_R)$, so the declaration $\hat{\alpha} : \kappa$ is in Γ_R. Since $\mathsf{FEV}(\tau) \subseteq \mathsf{dom}(\Gamma_R)$, the context Δ maintains the separation.

- **Case** InstReach: Here, $\Gamma = \Gamma_0[\hat{\alpha} : \kappa][\hat{\beta} : \kappa]$ and $\Delta = \Gamma_0[\hat{\alpha} : \kappa][\hat{\beta} : \kappa = \hat{\alpha}]$. We have $\hat{\alpha} \in \mathsf{dom}(\Gamma_R)$, so the declaration $\hat{\alpha} : \kappa$ is in Γ_R. Since $\hat{\beta}$ is declared to the right of $\hat{\alpha}$, it too must be in Γ_R, which can also be shown from $\mathsf{FEV}(\hat{\beta}) \subseteq \mathsf{dom}(\Gamma_R)$. Both declarations are in Γ_R, so the context Δ maintains the separation.

- **Case** InstZero: In this rule, Δ is the same as Γ except for a solution zero, which doesn't violate separation.

- **Case** InstSucc: The result follows by i.h., taking care to keep the declaration $\hat{\alpha}_1 : \mathbb{N}$ on the right when applying the i.h., even if $\hat{\alpha} : \mathbb{N}$ is the leftmost declaration in Γ_R, ensuring that $\mathsf{succ}(\hat{\alpha}_1)$ does not violate separation.

- **Case** InstBin: As in the InstSucc case, the new declarations should be kept on the right-hand side of the separator. Otherwise the case is straightforward (using the i.h. twice and transitivity).

Part (vi), propequivjudg: Similar to the CheckeqBin case of part (i), using the i.h. (i).
Part (vii), equivjudg:

- **Cases** $\equiv$Var, $\equiv$Exvar, $\equiv$Unit: Immediate ($\Delta_L = \Gamma_L$ and $\Delta_R = \Gamma_R$).
- **Case** $\equiv\oplus$: Similar to the case CheckeqBin in part (i).
- **Case** $\equiv$Vec: Similar to the case CheckeqBin in part (i).
- **Cases** $\equiv\forall$, $\equiv\exists$: Similar to the case CheckeqBin in part (i).
- **Cases** $\equiv\supset$, $\equiv\wedge$: Similar to the case CheckeqBin in part (i), using the i.h. (vi).
- **Cases** $\equiv$InstantiateL, $\equiv$InstantiateR: Use the i.h. (v). □

Lemma 71 (Separation for Subtyping). *If* $\Gamma_L * \Gamma_R \vdash A <:^{\pm} B \dashv \Delta$
and $\mathsf{FEV}(A) \subseteq \mathsf{dom}(\Gamma_R)$
and $\mathsf{FEV}(B) \subseteq \mathsf{dom}(\Gamma_R)$
then $\Delta = (\Delta_L * \Delta_R)$ *and* $(\Gamma_L * \Gamma_R) \xrightarrow{}_{*} (\Delta_L * \Delta_R)$.

Proof. By induction on the given derivation. In the <:Equiv case, use Lemma 70 (Separation for Auxiliary Judgments) (vii). Otherwise, the reasoning needed follows that used in the proof of Lemma 72 (Separation—Main). □

Lemma 72 (Separation—Main).

(Spines) *If* $\Gamma_L * \Gamma_R \vdash s : A\ p \gg C\ q \dashv \Delta$
or $\Gamma_L * \Gamma_R \vdash s : A\ p \gg C\ \lceil q \rceil \dashv \Delta$
and $\Gamma_L * \Gamma_R \vdash A\ p$ *type*
and $\mathsf{FEV}(A) \subseteq \mathsf{dom}(\Gamma_R)$
then $\Delta = (\Delta_L * \Delta_R)$ *and* $(\Gamma_L * \Gamma_R) \xrightarrow{}_{*} (\Delta_L * \Delta_R)$ *and* $\mathsf{FEV}(C) \subseteq \mathsf{dom}(\Delta_R)$.

(Checking) *If* $\Gamma_L * \Gamma_R \vdash e \Leftarrow C\ p \dashv \Delta$
and $\Gamma_L * \Gamma_R \vdash C\ p$ *type*
and $\mathsf{FEV}(C) \subseteq \mathsf{dom}(\Gamma_R)$
then $\Delta = (\Delta_L * \Delta_R)$ *and* $(\Gamma_L * \Gamma_R) \xrightarrow{}_{*} (\Delta_L * \Delta_R)$.

(Synthesis) *If* $\Gamma_L * \Gamma_R \vdash e \Rightarrow A\ p \dashv \Delta$
then $\Delta = (\Delta_L * \Delta_R)$ *and* $(\Gamma_L * \Gamma_R) \xrightarrow{}_{*} (\Delta_L * \Delta_R)$.

(Match) *If* $\Gamma_L * \Gamma_R \vdash \Pi :: \vec{A} \Leftarrow C\ p \dashv \Delta$
and $\mathsf{FEV}(\vec{A}) = \emptyset$
and $\mathsf{FEV}(C) \subseteq \mathsf{dom}(\Gamma_R)$
then $\Delta = (\Delta_L * \Delta_R)$ *and* $(\Gamma_L * \Gamma_R) \xrightarrow{}_{*} (\Delta_L * \Delta_R)$.

(Match Elim.) *If* $\Gamma_L * \Gamma_R \,/\, P \vdash \Pi :: \vec{A} \Leftarrow C\ p \dashv \Delta$
and $\mathsf{FEV}(P) = \emptyset$
and $\mathsf{FEV}(\vec{A}) = \emptyset$
and $\mathsf{FEV}(C) \subseteq \mathsf{dom}(\Gamma_R)$
then $\Delta = (\Delta_L * \Delta_R)$ *and* $(\Gamma_L * \Gamma_R) \xrightarrow{}_{*} (\Delta_L * \Delta_R)$.

Proof. By induction on the given derivation.
First, the (Match) judgment part, giving only the cases that motivate the side conditions:

- **Case** MatchBase: Here we use the i.h. (Checking), for which we need $\mathsf{FEV}(C) \subseteq \mathsf{dom}(\Gamma_R)$.
- **Case** Match$\wedge$: Here we use the i.h. (Match Elim.), which requires that $\mathsf{FEV}(P) = \emptyset$, which motivates $\mathsf{FEV}(\vec{A}) = \emptyset$.
- **Case** MatchNeg: In its premise, this rule appends a type $A \in \vec{A}$ to Γ_R and claims it is principal ($z : A!$), which motivates $\mathsf{FEV}(\vec{A} = \emptyset)$.

Similarly, (Match Elim.):

- **Case** MatchUnify: Here we use Lemma 70 (Separation for Auxiliary Judgments) (iii), for which we need $\mathsf{FEV}(\sigma) \cup \mathsf{FEV}(\tau) = \emptyset$, which motivates $\mathsf{FEV}(P) = \emptyset$.

Now, we show the cases for the (Spine), (Checking), and (Synthesis) parts.

- **Cases** Var, 1I, $\supset$I$\bot$: In all of these rules, the output context is the same as the input context, so just let $\Delta_L = \Gamma_L$ and $\Delta_R = \Gamma_R$.

- **Case**

$$\frac{}{\Gamma_L * \Gamma_R \vdash \cdot : A\ p \gg \underbrace{A}_{C}\ \underbrace{p}_{q} \dashv \Gamma_L * \Gamma_R}\ \mathsf{EmptySpine}$$

Let $\Delta_L = \Gamma_L$ and $\Delta_R = \Gamma_R$.
We have $\mathsf{FEV}(A) \subseteq \mathsf{dom}(\Gamma_R)$. Since $\Delta_R = \Gamma_R$ and $C = A$, it is immediate that $\mathsf{FEV}(C) \subseteq \mathsf{dom}(\Delta_R)$.

- **Case**

$$\frac{\Gamma_L * \Gamma_R \vdash e \Rightarrow A\ q \dashv \Theta \qquad \Theta \vdash A <:^* B \dashv \Delta}{\Gamma_L * \Gamma_R \vdash e \Leftarrow B\ p \dashv \Delta}\ \mathsf{Sub}$$

By i.h., $\Theta = (\Theta_L * \Theta_R)$ and $(\Gamma_L * \Gamma_R) \xrightarrow{}{}_{*} (\Theta_L * \Theta_R)$.
By Lemma 71 (Separation for Subtyping), $\Delta = (\Delta_L * \Delta_R)$ and $(\Theta_L * \Theta_R) \xrightarrow{}{}_{*} (\Delta_L * \Delta_R)$.
By Lemma 68 (Transitivity of Separation), $(\Gamma_L * \Gamma_R) \xrightarrow{}{}_{*} (\Delta_L * \Delta_R)$.

- **Case**

$$\frac{\Gamma \vdash A\,!\ \mathit{type} \qquad \Gamma \vdash e \Leftarrow [\Gamma]A\ !\ \dashv \Delta}{\Gamma \vdash (e : A) \Rightarrow [\Delta]A\ !\ \dashv \Delta}\ \mathsf{Anno}$$

By i.h.; since $\mathsf{FEV}(A) = \emptyset$, the condition on the (Checking) part is trivial.

- **Case**

$$\frac{}{\Gamma[\hat{\alpha} : \star] \vdash ()\ \Leftarrow \hat{\alpha}\ \dashv \Gamma[\hat{\alpha} : \star = 1]}\ 1\mathsf{I}\hat{\alpha}$$

Adding a solution with a ground type cannot destroy separation.

- **Case**

$$\frac{\nu\ \mathit{chk\text{-}I} \qquad \Gamma_L, \Gamma_R, \alpha : \kappa \vdash \nu \Leftarrow A_0\ p \dashv \Delta, \alpha : \kappa, \Theta}{\Gamma_L, \Gamma_R \vdash \nu \Leftarrow \forall \alpha : \kappa.\, A_0\ p \dashv \Delta}\ \forall\mathsf{I}$$

	$\mathsf{FEV}(\forall \alpha : \kappa.\, A_0) \subseteq \mathsf{dom}(\Gamma_R)$		Given
	$\mathsf{FEV}(A_0) \subseteq \mathsf{dom}(\Gamma_R, \alpha : \kappa)$		From definition of FEV
	$(\Delta, \alpha : \kappa, \Theta) = (\Delta_L * \Delta'_R)$		By i.h.
	$(\Gamma_L * (\Gamma_R, \alpha : \kappa)) \xrightarrow{}{}_{*} (\Delta_L * \Delta'_R)$		$''$
☞	$(\Gamma_L * \Gamma_R) \xrightarrow{}{}_{*} (\Delta_L * \Delta_R)$		By Lemma 69 (Separation Truncation)
	$\Delta'_R = (\Delta_R, \alpha : \kappa, \Theta)$		$''$
	$(\Delta, \alpha : \kappa, \Theta) = (\Delta_L * \Delta'_R)$		Above
		$= (\Delta_L, \Delta'_R)$	Definition of $*$
		$= (\Delta_L, \Delta_R, \alpha : \kappa, \Theta)$	By above equation
☞	$\Delta = (\Delta_L, \Delta_R)$		α not multiply declared

- **Case**

$$\frac{\Gamma_L, \Gamma_R, \hat{\alpha} : \kappa \vdash e\ s : [\hat{\alpha}/\alpha]A_0 \gg C\ q \dashv \Delta}{\Gamma_L, \Gamma_R \vdash e\ s : \forall \alpha : \kappa.\, A_0\ p \gg C\ q \dashv \Delta}\ \forall\mathsf{Spine}$$

	$\mathsf{FEV}(\forall\alpha:\kappa A_0.\,) \subseteq \mathsf{dom}(\Gamma_R)$	Given
	$\mathsf{FEV}([\hat{\alpha}/\alpha]A_0) \subseteq \mathsf{dom}(\Gamma_R, \hat{\alpha}:\kappa)$	From definition of FEV
☞	$\Delta = (\Delta_L * \Delta_R)$	By i.h.
	$(\Gamma_L * (\Gamma_R, \hat{\alpha}:\kappa)) \xrightarrow[*]{} (\Delta_L * \Delta_R)$	$''$
☞	$\mathsf{FEV}(C) \subseteq \mathsf{dom}(\Delta_R)$	$''$
	$\mathsf{dom}(\Gamma_L) \subseteq \mathsf{dom}(\Delta_L)$	By Definition 5
	$\mathsf{dom}(\Gamma_R, \hat{\alpha}:\kappa) \subseteq \mathsf{dom}(\Delta_R)$	By Definition 5
	$\mathsf{dom}(\Gamma_R) \cup \{\hat{\alpha}\} \subseteq \mathsf{dom}(\Delta_R)$	By definition of $\mathsf{dom}(-)$
	$\mathsf{dom}(\Gamma_R) \subseteq \mathsf{dom}(\Delta_R)$	Property of $\subseteq$
	$(\Gamma_L, \Gamma_R) \longrightarrow (\Delta_L, \Delta_R)$	By Lemma 51 (Typing Extension)
☞	$(\Gamma_L * \Gamma_R) \xrightarrow[*]{} (\Delta_L * \Delta_R)$	By Definition 5

- **Case** $\dfrac{e \text{ not a case} \qquad \Gamma_L * \Gamma_R \vdash P\ \mathit{true} \dashv \Theta \qquad \Theta \vdash e \Leftarrow [\Theta]A_0\ p \dashv \Delta}{\Gamma_L * \Gamma_R \vdash e \Leftarrow (A_0 \wedge P)\ p \dashv \Delta}\ \wedge\mathsf{I}$

	$\Gamma_L * \Gamma_R \vdash (A_0 \wedge P)\ p\ \mathit{type}$	Given
	$\Gamma_L * \Gamma_R \vdash P\ \mathit{prop}$	By inversion
	$\Gamma_L * \Gamma_R \vdash A_0\ p\ \mathit{type}$	By inversion
	$\mathsf{FEV}(A_0 \wedge P) \subseteq \mathsf{dom}(\Gamma_R)$	Given
	$\mathsf{FEV}(P) \subseteq \mathsf{dom}(\Gamma_R)$	By def. of FEV
	$\mathsf{FEV}(A_0) \subseteq \mathsf{dom}(\Gamma_R)$	$''$
	$\Theta = (\Theta_L * \Theta_R)$	By Lemma 70 (Separation for Auxiliary Judgments) (i)
	$(\Gamma_L * \Gamma_R) \xrightarrow[*]{} (\Theta_L * \Theta_R)$	$''$
	$\mathsf{FEV}(A_0) \subseteq \mathsf{dom}(\Gamma_R)$	Above
	$\mathsf{dom}(\Gamma_R) \subseteq \mathsf{dom}(\Theta_R)$	By Definition 5
	$\mathsf{FEV}(A_0) \subseteq \mathsf{dom}(\Theta_R)$	By previous line
	$\mathsf{FEV}([\Theta]A_0) \subseteq \mathsf{dom}(\Theta_R)$	Previous line and $(\Gamma_L * \Gamma_R) \xrightarrow[*]{} (\Theta_L * \Theta_R)$
	$\Gamma_L * \Gamma_R \vdash (A_0 \wedge P)\ p\ \mathit{type}$	Given
	$\Gamma_L * \Gamma_R \vdash A_0\ p\ \mathit{type}$	By inversion
	$\Theta \vdash A_0\ p\ \mathit{type}$	By Lemma 41 (Extension Weakening for Principal Typing)
	$\Theta \vdash [\Theta]A_0\ p\ \mathit{type}$	By Lemma 13 (Right-Hand Substitution for Typing)
☞	$\Delta = (\Delta_L * \Delta_R)$	By i.h.
	$(\Theta_L * \Theta_R) \xrightarrow[*]{} (\Delta_L * \Delta_R)$	$''$
☞	$(\Gamma_L * \Gamma_R) \xrightarrow[*]{} (\Delta_L * \Delta_R)$	By Lemma 68 (Transitivity of Separation)

- **Case** Nil: Similar to a section of the $\wedge\mathsf{I}$ case.
- **Case** Cons: Similar to the $\wedge\mathsf{I}$ case, with an extra use of the i.h. for the additional second premise.
- **Case** $\dfrac{v\ \mathit{chk\text{-}I} \qquad \Gamma_L * (\Gamma_R, \blacktriangleright_P) \,/\, P \dashv \Theta \qquad \Theta \vdash v \Leftarrow [\Theta]A_0\ !\ \dashv \Delta, \blacktriangleright_P, \Delta'}{\Gamma_L * \Gamma_R \vdash v \Leftarrow P \supset A_0\ !\ \dashv \Delta}\ \supset\mathsf{I}$

$\Gamma_L * \Gamma_R \vdash (P \supset A_0)\ \mathbf{!}\ \mathit{type}$ Given
$\Gamma_L * \Gamma_R \vdash P \supset A_0\ \mathit{prop}$ By inversion
$\mathsf{FEV}(P \supset A_0) = \emptyset$ ″
$\mathsf{FEV}(P) = \emptyset$ By def. of FEV

$\Gamma_L * (\Gamma_R, \blacktriangleright_P)\ /\ P \dashv \Theta$ Subderivation
$\Theta = (\Theta_L * (\Theta_R, \Theta_Z))$ By Lemma 70 (Separation for Auxiliary Judgments) (iv)
$(\Gamma_L * (\Gamma_R, \blacktriangleright_P, \Theta_Z)) \xrightarrow[*]{} (\Theta_L * (\Theta_R, \Theta_Z))$ ″

$\Gamma_L * \Gamma_R \vdash (P \supset A_0)\ \mathbf{!}\ \mathit{type}$ Given
$\Gamma_L, \Gamma_R \vdash A_0\ \mathbf{!}\ \mathit{type}$ By Lemma 42 (Inversion of Principal Typing) (2)
$\Gamma_L, \Gamma_R, \blacktriangleright_P, \Theta_Z \vdash A_0\ \mathbf{!}\ \mathit{type}$ By Lemma 35 (Suffix Weakening)
$\Theta \vdash [\Theta]A_0\ \mathbf{!}\ \mathit{type}$ By Lemmas 41 and 40
$\mathsf{FEV}(A_0) = \emptyset$ Above and def. of FEV
$\mathsf{FEV}(A_0) \subseteq \mathsf{dom}(\Theta_R, \Theta_Z)$ Immediate
$(\Delta, \blacktriangleright_P, \Delta') = (\Delta_L * \Delta'_R)$ By i.h.
$(\Theta_L * (\Theta_R, \Theta_Z)) \xrightarrow[*]{} (\Delta_L * \Delta'_R)$ ″

$(\Gamma_L * (\Gamma_R, \blacktriangleright_P)) \xrightarrow[*]{} (\Delta_L * \Delta'_R)$ By Lemma 68 (Transitivity of Separation)
☞ $(\Gamma_L * \Gamma_R) \xrightarrow[*]{} (\Delta_L * \Delta_R)$ By Lemma 69 (Separation Truncation)
$\Delta'_R = (\Delta_R, \blacktriangleright_P, \dots)$ ″
☞ $\Delta = (\Delta_L, \Delta_R)$ Similar to the ∀I case

- **Case** $\dfrac{\Gamma_L * \Gamma_R \vdash P\ \mathit{true} \dashv \Theta \qquad \Theta \vdash e\ s : [\Theta]A_0\ p \gg C\ q \dashv \Delta}{\Gamma_L * \Gamma_R \vdash e\ s : P \supset A_0\ p \gg C\ q \dashv \Delta}\ \supset\mathsf{Spine}$

$\Gamma_L * \Gamma_R \vdash (P \supset A_0)\ p\ \mathit{type}$ Given
$\Gamma_L * \Gamma_R \vdash P\ \mathit{prop}$ By inversion
$\Gamma_L, \Gamma_R \vdash P\ \mathit{true} \dashv \Theta$ Subderivation
$\Theta = (\Theta_L * \Theta_R)$ By Lemma 70 (Separation for Auxiliary Judgments) (i)
$(\Gamma_L * \Gamma_R) \xrightarrow[*]{} (\Theta_L * \Theta_R)$ ″
$\Theta \vdash e\ s : [\Theta]A_0\ p \gg C\ q \dashv \Delta$ Subderivation
$(\Delta, \blacktriangleright_P, \Delta') = (\Delta_L * \Delta'_R)$ By i.h.
$(\Theta_L * \Theta_R) \xrightarrow[*]{} (\Delta_L * \Delta'_R)$ ″
☞ $\mathsf{FEV}(C) \subseteq \mathsf{dom}(\Delta_R)$ ″
☞ $(\Gamma_L * \Gamma_R) \xrightarrow[*]{} (\Delta_L * \Delta_R)$ By Lemma 68 (Transitivity of Separation)

- **Case** $\dfrac{\Gamma_L, \Gamma_R, x : C\ p \vdash v \Leftarrow C\ p \dashv \Delta, x : C\ p, \Theta}{\Gamma_L, \Gamma_R \vdash \mathsf{rec}\ x.v \Leftarrow C\ p \dashv \Delta}\ \mathsf{Rec}$

$\Gamma_L * \Gamma_R \vdash C\ p\ \mathit{type}$ Given
$\mathsf{FEV}(C) \subseteq \mathsf{dom}(\Gamma_R)$ Given
$\Gamma_L * (\Gamma_R, x : C\ p) \vdash C\ p\ \mathit{type}$ By weakening and Definition 4
$\Gamma_L, \Gamma_R, x : C\ p \vdash v \Leftarrow C\ p \dashv \Delta, x : C\ p, \Theta$ Subderivation
$(\Delta, x : C\ p, \Theta) = (\Delta_L, \Delta'_R)$ By i.h.
$(\Gamma_L * \Gamma_R) \xrightarrow[*]{} (\Delta_L * \Delta'_R)$ ″
☞ $(\Gamma_L * \Gamma_R) \xrightarrow[*]{} (\Delta_L * \Delta_R)$ By Lemma 69 (Separation Truncation)
$\Delta'_R = (\Delta_R, x : C\ p, \dots)$ ″
☞ $\Delta = (\Delta_L, \Delta_R)$ Similar to the ∀I case

- **Case** $\dfrac{\Gamma_L, \Gamma_R, x : A\ p \vdash e \Leftarrow B\ p \dashv \Delta, x : A\ p, \Theta}{\Gamma_L, \Gamma_R \vdash \lambda x.e \Leftarrow A \to B\ p \dashv \Delta}\ \to\mathsf{I}$

	$\Gamma_L * \Gamma_R \vdash (A \to B)\ p\ \mathit{type}$	Given
	$\Gamma_L * \Gamma_R \vdash B\ p\ \mathit{type}$	By inversion
	$\mathsf{FEV}(A \to B) \subseteq \mathsf{dom}(\Gamma_R)$	Given
	$\mathsf{FEV}(A) \subseteq \mathsf{dom}(\Gamma_R)$	By def. of FEV
	$\Gamma_L * (\Gamma_R, x : A\ p) \vdash B\ p\ \mathit{type}$	By weakening and Definition 4
	$\Gamma_L, \Gamma_R, x : A\ p \vdash e \Leftarrow B\ p \dashv \Delta, x : A\ p, \Theta$	Subderivation
	$(\Delta, x : A\ p, \Theta) = (\Delta_L, \Delta'_R)$	By i.h.
	$(\Gamma_L * \Gamma_R) \xrightarrow[*]{} (\Delta_L * \Delta'_R)$	$''$
☞	$(\Gamma_L * \Gamma_R) \xrightarrow[*]{} (\Delta_L * \Delta_R)$	By Lemma 69 (Separation Truncation)
	$\Delta'_R = (\Delta_R, x : A\ p, \dots)$	$''$
☞	$\Delta = (\Delta_L, \Delta_R)$	Similar to the $\forall$I case

- **Case** $\dfrac{\Gamma_0[\hat\alpha_1{:}\star, \hat\alpha_2{:}\star, \hat\alpha{:}\star = \hat\alpha_1 \to \hat\alpha_2], x : \hat\alpha_1 \vdash e_0 \Leftarrow \hat\alpha_2 \dashv \Delta, x : \hat\alpha_1, \Delta'}{\underbrace{\Gamma_0[\hat\alpha : \star]}_{\Gamma_L * \Gamma_R} \vdash \lambda x.\, e_0 \Leftarrow \hat\alpha \dashv \Delta}\ {\to}\mathsf{I}\hat\alpha$

  We have $(\Gamma_L * \Gamma_R) = \Gamma_0[\hat\alpha : \star]$. We also have $\mathsf{FEV}(\hat\alpha) \subseteq \mathsf{dom}(\Gamma_R)$. Therefore $\hat\alpha \in \mathsf{dom}(\Gamma_R)$ and

  $$\Gamma_0[\hat\alpha : \star] \quad = \quad \Gamma_L, \Gamma_2, \hat\alpha : \star, \Gamma_3$$

  where $\Gamma_R = (\Gamma_2, \hat\alpha : \star, \Gamma_3)$.

  Then the input context in the premise has the following form:

  $$\Gamma_0[\hat\alpha_1{:}\star, \hat\alpha_2{:}\star, \hat\alpha{:}\star = \hat\alpha_1 \to \hat\alpha_2], x : \hat\alpha_1 \quad = \quad \Gamma_L, \Gamma_2, \hat\alpha_1{:}\star, \hat\alpha_2{:}\star, \hat\alpha{:}\star = \hat\alpha_1 \to \hat\alpha_2, \Gamma_3, x : \hat\alpha_1$$

  Let us separate this context at the same point as $\Gamma_0[\hat\alpha : \star]$, that is, after Γ_L and before Γ_2, and call the resulting right-hand context Γ'_R. That is,

  $$\Gamma_0[\hat\alpha_1{:}\star, \hat\alpha_2{:}\star, \hat\alpha{:}\star = \hat\alpha_1 \to \hat\alpha_2], x : \hat\alpha_1 \quad = \quad \Gamma_L * \big(\underbrace{\Gamma_2, \hat\alpha_1{:}\star, \hat\alpha_2{:}\star, \hat\alpha{:}\star = \hat\alpha_1 \to \hat\alpha_2, \Gamma_3, x : \hat\alpha_1}_{\Gamma'_R}\big)$$

	$\mathsf{FEV}(\hat\alpha) \subseteq \mathsf{dom}(\Gamma_R)$	Given
	$\Gamma_L * \Gamma'_R \vdash e_0 \Leftarrow \hat\alpha_2 \dashv \Delta, x : \hat\alpha_1, \Delta'$	Subderivation
	$\Gamma_L * \Gamma'_R \vdash \hat\alpha_2\ \not{!}\ \mathit{type}$	$\hat\alpha_2 \in \mathsf{dom}(\Gamma'_R)$
	$\mathsf{FEV}(\hat\alpha_2) \subseteq \mathsf{dom}(\Gamma'_R)$	$\hat\alpha_2 \in \mathsf{dom}(\Gamma'_R)$
	$(\Delta, x : \hat\alpha_1, \Delta') = (\Delta_L, \Delta'_R)$	By i.h.
	$(\Gamma_L * \Gamma'_R) \xrightarrow[*]{} (\Delta_L * \Delta'_R)$	$''$
☞	$\Delta = (\Delta_L, \Delta_R)$	Similar to the $\forall$I case
☞	$(\Gamma_L * \Gamma_R) \xrightarrow[*]{} (\Delta_L * \Delta_R)$	$''$

- **Case** $\dfrac{\Gamma \vdash e \Rightarrow A\ p \dashv \Theta \qquad \Theta \vdash s : [\Theta]A\ p \gg C\ \lceil q \rceil \dashv \Delta}{\Gamma \vdash e\ s \Rightarrow C\ q \dashv \Delta}\ {\to}\mathsf{E}$

  Use the i.h. and Lemma 68 (Transitivity of Separation), with Lemma 89 (Well-formedness of Algorithmic Typing) and Lemma 13 (Right-Hand Substitution for Typing).

- **Case** $\dfrac{\Gamma \vdash s : A\ \mathbf{!} \gg C\ \not{!} \dashv \Delta \qquad \mathsf{FEV}([\Delta]C) = \emptyset}{\Gamma \vdash s : A\ \mathbf{!} \gg C\ \lceil \mathbf{!} \rceil \dashv \Delta}\ \mathsf{SpineRecover}$

  Use the i.h.

- **Case** $\dfrac{\Gamma \vdash s : A\ p \gg C\ q \dashv \Delta \qquad \big((p = \not{!}) \ \text{ or } \ (q = \mathbf{!}) \ \text{ or } \ (\mathsf{FEV}([\Delta]C) \neq \emptyset)\big)}{\Gamma \vdash s : A\ p \gg C\ \lceil q \rceil \dashv \Delta}\ \mathsf{SpinePass}$

  Use the i.h.

- **Case** $\dfrac{\Gamma_L * \Gamma_R \vdash e \Leftarrow A_1\ p \dashv \Theta \qquad \Theta \vdash s : [\Theta]A_2\ p \gg C\ q \dashv \Delta}{\Gamma_L * \Gamma_R \vdash e\ s : A_1 \to A_2\ p \gg C\ q \dashv \Delta} \to\mathsf{Spine}$

	$\Gamma \vdash (A_1 \to A_2)\ p\ \mathit{type}$	Given
	$\Gamma \vdash A_1\ \mathit{type}$	By inversion
	$\mathsf{FEV}(A_1 \to A_2) \subseteq \mathsf{dom}(\Gamma_R)$	Given
	$\mathsf{FEV}(A_1) \subseteq \mathsf{dom}(\Gamma_R)$	By def. of FEV
	$\Theta = (\Theta_L, \Theta_R)$	By i.h.
	$(\Gamma_L * \Gamma_R) \xrightarrow[*]{} (\Theta_L * \Theta_R)$	$''$
	$\Gamma \vdash A_2\ \mathit{type}$	By inversion
	$\Gamma \vdash [\Theta]A_2\ \mathit{type}$	By Lemma 13 (Right-Hand Substitution for Typing)
	$\mathsf{FEV}(A_2) \subseteq \mathsf{dom}(\Gamma_R)$	By def. of FEV
☞	$\Delta = (\Delta_L, \Delta_R)$	By i.h.
	$(\Theta_L * \Theta_R) \xrightarrow[*]{} (\Delta_L * \Delta_R)$	$''$
☞	$\mathsf{FEV}(C) \subseteq \mathsf{dom}(\Delta_R)$	$''$
☞	$(\Gamma_L * \Gamma_R) \xrightarrow[*]{} (\Delta_L * \Delta_R)$	By Lemma 68 (Transitivity of Separation)

- **Case** $\dfrac{\Gamma \vdash e \Leftarrow A_k\ p \dashv \Delta}{\Gamma \vdash \mathsf{inj}_k\ e \Leftarrow A_1 + A_2\ p \dashv \Delta} +\mathsf{I}_k$

  Use the i.h. (inverting $\Gamma \vdash (A_1 + A_2)\ p\ \mathit{type}$).

- **Case** $\dfrac{\Gamma \vdash e_1 \Leftarrow A_1\ p \dashv \Theta \qquad \Theta \vdash e_2 \Leftarrow [\Theta]A_2\ p \dashv \Delta}{\Gamma \vdash \langle e_1, e_2 \rangle \Leftarrow A_1 \times A_2\ p \dashv \Delta} \times\mathsf{I}$

	$\Gamma \vdash (A_1 \times A_2)\ p\ \mathit{type}$	Given
	$\Gamma \vdash A_1\ p\ \mathit{type}$	By inversion
	$\Gamma \vdash e_1 \Leftarrow A_1\ p \dashv \Theta$	Subderivation
	$\Theta = (\Theta_L, \Theta_R)$	By i.h.
	$(\Gamma_L * \Gamma_R) \xrightarrow[*]{} (\Theta_L * \Theta_R)$	$''$
	$\Gamma \vdash A_2\ \mathit{type}$	By inversion
	$\Gamma \longrightarrow \Theta$	By Lemma 51 (Typing Extension)
	$\Theta \vdash A_2\ \mathit{type}$	By Lemma 36 (Extension Weakening (Sorts))
	$\Theta \vdash [\Theta]A_2\ \mathit{type}$	By Lemma 13 (Right-Hand Substitution for Typing)
	$\Theta \vdash e_2 \Leftarrow [\Theta]A_2\ p \dashv \Delta$	Subderivation
	$\Delta = (\Delta_L, \Delta_R)$	By i.h.
	$(\Theta_L * \Theta_R) \xrightarrow[*]{} (\Delta_L * \Delta_R)$	$''$
☞	$(\Gamma_L * \Gamma_R) \xrightarrow[*]{} (\Delta_L * \Delta_R)$	By Lemma 68 (Transitivity of Separation)

- **Case** $\dfrac{\Gamma[\hat\alpha_2{:}\star, \hat\alpha_1{:}\star, \hat\alpha{:}\star = \hat\alpha_1 \times \hat\alpha_2] \vdash e_1 \Leftarrow \hat\alpha_1 \dashv \Theta \qquad \Theta \vdash e_2 \Leftarrow [\Theta]\hat\alpha_2 \dashv \Delta}{\Gamma[\hat\alpha : \star] \vdash \langle e_1, e_2\rangle \Leftarrow \hat\alpha \dashv \Delta}\ \times I\hat\alpha$

  We have $(\Gamma_L * \Gamma_R) = \Gamma_0[\hat\alpha : \star]$. We also have $\mathsf{FEV}(\hat\alpha) \subseteq \mathsf{dom}(\Gamma_R)$. Therefore $\hat\alpha \in \mathsf{dom}(\Gamma_R)$ and

  $$\Gamma_0[\hat\alpha : \star] \quad = \quad \Gamma_L, \Gamma_2, \hat\alpha : \star, \Gamma_3$$

  where $\Gamma_R = (\Gamma_2, \hat\alpha : \star, \Gamma_3)$.

  Then the input context in the premise has the following form:

  $$\Gamma_0[\hat\alpha_1{:}\star, \hat\alpha_2{:}\star, \hat\alpha{:}\star = \hat\alpha_1 \times \hat\alpha_2] \quad = \quad (\Gamma_L, \Gamma_2, \hat\alpha_1{:}\star, \hat\alpha_2{:}\star, \hat\alpha{:}\star = \hat\alpha_1 \times \hat\alpha_2, \Gamma_3)$$

  Let us separate this context at the same point as $\Gamma_0[\hat\alpha : \star]$, that is, after Γ_L and before Γ_2, and call the resulting right-hand context Γ'_R:

  $$\Gamma_0[\hat\alpha_1{:}\star, \hat\alpha_2{:}\star, \hat\alpha{:}\star = \hat\alpha_1 \times \hat\alpha_2] \quad = \quad \Gamma_L \ * \ \big(\underbrace{\Gamma_2, \hat\alpha_1{:}\star, \hat\alpha_2{:}\star, \hat\alpha{:}\star = \hat\alpha_1 \times \hat\alpha_2, \Gamma_3}_{\Gamma'_R}\big)$$

	$\mathsf{FEV}(\hat\alpha) \subseteq \mathsf{dom}(\Gamma_R)$	Given
	$\Gamma_L * \Gamma'_R \vdash e_1 \Leftarrow \hat\alpha_1 \dashv \Theta$	Subderivation
	$\mathsf{FEV}(\hat\alpha_2) \subseteq \mathsf{dom}(\Gamma'_R)$	$\hat\alpha_2 \in \mathsf{dom}(\Gamma'_R)$
	$\Theta = (\Theta_L, \Theta_R)$	By i.h.
	$(\Gamma_L * \Gamma'_R) \longrightarrow\!\!\!\!\!* \ (\Theta_L * \Theta_R)$	$''$
	$\Theta \vdash e_2 \Leftarrow [\Theta]\hat\alpha_2 \dashv \Delta$	Subderivation
	$\mathsf{dom}(\Gamma'_R) \subseteq \mathsf{dom}(\Theta_R)$	By Definition 5
	$\mathsf{FEV}(\hat\alpha_2) \subseteq \mathsf{dom}(\Theta_R)$	By above $\subseteq$
	$\mathsf{FEV}([\Theta_R]\hat\alpha_2) \subseteq \mathsf{dom}(\Theta_R)$	By Definition 4
☞	$\Delta = (\Delta_L, \Delta_R)$	By i.h.
	$(\Theta_L * \Theta_R) \longrightarrow\!\!\!\!\!* \ (\Delta_L * \Delta_R)$	$''$

$\Gamma_R = (\Gamma_2, \hat\alpha : \star, \Gamma_3)$	Above
$\Gamma'_R = (\Gamma_2, \hat\alpha_1{:}\star, \hat\alpha_2{:}\star, \hat\alpha : \star = \hat\alpha_1 \times \hat\alpha_2, \Gamma_3)$	Above

  By Lemma 23 (Deep Evar Introduction) (i), (i), (ii) and the definition of separation, we can show

  $(\Gamma_L * (\Gamma_2, \hat\alpha : \star, \Gamma_3)) \longrightarrow\!\!\!\!\!* \ (\Gamma_L * (\Gamma_2, \hat\alpha_1{:}\star, \hat\alpha_2{:}\star, \hat\alpha{:}\star = \hat\alpha_1 \times \hat\alpha_2, \Gamma_3))$

  $(\Gamma_L * \Gamma_R) \longrightarrow\!\!\!\!\!* \ (\Gamma_L * \Gamma'_R)$ By above equalities

  ☞ $(\Gamma_L * \Gamma_R) \longrightarrow\!\!\!\!\!* \ (\Delta_L * \Delta_R)$ By Lemma 68 (Transitivity of Separation) twice

- **Case** $\dfrac{\Gamma[\hat\alpha_1 : \star, \hat\alpha_2 : \star, \hat\alpha : \star = \hat\alpha_1 + \hat\alpha_2] \vdash e \Leftarrow \hat\alpha_k \dashv \Delta}{\Gamma[\hat\alpha : \star] \vdash \mathsf{inj}_k\, e \Leftarrow \hat\alpha \dashv \Delta}\ +I\hat\alpha_k$

  Similar to the $\times I\hat\alpha$ case, but simpler.

- **Case** $\dfrac{\Gamma[\hat\alpha_2 : \star, \hat\alpha_1 : \star, \hat\alpha : \star = \hat\alpha_1 \to \hat\alpha_2] \vdash e\ s_0 : (\hat\alpha_1 \to \hat\alpha_2) \gg C \dashv \Delta}{\Gamma[\hat\alpha : \star] \vdash e\ s_0 : \hat\alpha \gg C \dashv \Delta}\ \hat\alpha\mathsf{Spine}$

  Similar to the $\times I\hat\alpha$ and $+I\hat\alpha_k$ cases, except that (because we're in the spine part of the lemma) we have to show that $\mathsf{FEV}(C) \subseteq \mathsf{dom}(\Delta_R)$. But we have the same C in the premise and conclusion, so we get that by applying the i.h.

- **Case** $\dfrac{\Gamma \vdash e \Rightarrow A\ ! \dashv \Theta \qquad \Theta \vdash \Pi :: A \Leftarrow [\Theta]C\ p \dashv \Delta \qquad \Delta \vdash \Pi\ \mathit{covers}\ [\Delta]A}{\Gamma \vdash \mathsf{case}(e, \Pi) \Leftarrow C\ p \dashv \Delta}\ \mathsf{Case}$

  Use the i.h. and Lemma 68 (Transitivity of Separation). □

I′ Decidability of Algorithmic Subtyping

I′.1 Lemmas for Decidability of Subtyping

Lemma 73 (Substitution Isn't Large)**.**
For all contexts Θ*, we have* $\#\mathsf{large}([\Theta]A) = \#\mathsf{large}(A)$.

Proof. By induction on A, following the definition of substitution. □

Lemma 74 (Instantiation Solves)**.**
If $\Gamma \vdash \hat{\alpha} := \tau : \kappa \dashv \Delta$ *and* $[\Gamma]\tau = \tau$ *and* $\hat{\alpha} \notin \mathrm{FV}([\Gamma]\tau)$ *then* $|\mathsf{unsolved}(\Gamma)| = |\mathsf{unsolved}(\Delta)| + 1$.

Proof. By induction on the given derivation.

- **Case**
$$\frac{\Gamma_L \vdash \tau : \kappa}{\Gamma_L, \hat{\alpha} : \kappa, \Gamma_R \vdash \hat{\alpha} := \tau : \kappa \dashv \Gamma_L, \hat{\alpha} : \kappa = \tau, \Gamma_R}\ \mathsf{InstSolve}$$
It is evident that $|\mathsf{unsolved}(\Gamma_L, \hat{\alpha} : \kappa, \Gamma_R)| = |\mathsf{unsolved}(\Gamma_L, \hat{\alpha} : \kappa = \tau, \Gamma_R)| + 1$.

- **Case**
$$\frac{\hat{\beta} \in \mathsf{unsolved}(\Gamma[\hat{\alpha} : \kappa][\hat{\beta} : \kappa])}{\Gamma[\hat{\alpha} : \kappa][\hat{\beta} : \kappa] \vdash \hat{\alpha} := \underbrace{\hat{\beta}}_{\tau} : \kappa \dashv \Gamma[\hat{\alpha} : \kappa][\hat{\beta} : \kappa = \hat{\alpha}]}\ \mathsf{InstReach}$$
Similar to the previous case.

- **Case**
$$\frac{\Gamma_0[\hat{\alpha}_2 : \star, \hat{\alpha}_1 : \star, \hat{\alpha} : \star = \hat{\alpha}_1 \oplus \hat{\alpha}_2] \vdash \hat{\alpha}_1 := \tau_1 : \star \dashv \Theta \qquad \Theta \vdash \hat{\alpha}_2 := [\Theta]\tau_2 : \star \dashv \Delta}{\Gamma_0[\hat{\alpha} : \star] \vdash \hat{\alpha} := \tau_1 \oplus \tau_2 : \star \dashv \Delta}\ \mathsf{InstBin}$$

$$\begin{aligned}
|\mathsf{unsolved}(\Gamma_0[\hat{\alpha}_2 : \star, \hat{\alpha}_1 : \star, \hat{\alpha} = \hat{\alpha}_1 \oplus \hat{\alpha}_2])| &= |\mathsf{unsolved}(\Gamma_0[\hat{\alpha}])| + 1 && \text{Immediate}\\
|\mathsf{unsolved}(\Gamma_0[\hat{\alpha}_2 : \star, \hat{\alpha}_1 : \star, \hat{\alpha} = \hat{\alpha}_1 \oplus \hat{\alpha}_2])| &= |\mathsf{unsolved}(\Theta)| + 1 && \text{By i.h.}\\
|\mathsf{unsolved}(\Gamma)| &= |\mathsf{unsolved}(\Theta)| && \text{Subtracting 1}\\
\text{☞} \qquad &= |\mathsf{unsolved}(\Delta)| + 1 && \text{By i.h.}
\end{aligned}$$

- **Case**
$$\frac{}{\Gamma[\hat{\alpha} : \mathbb{N}] \vdash \hat{\alpha} := \mathsf{zero} : \mathbb{N} \dashv \Gamma[\hat{\alpha} : \mathbb{N} = \mathsf{zero}]}\ \mathsf{InstZero}$$
Similar to the InstSolve case.

- **Case**
$$\frac{\Gamma_0[\hat{\alpha}_1 : \mathbb{N}, \hat{\alpha} : \mathbb{N} = \mathsf{succ}(\hat{\alpha}_1)] \vdash \hat{\alpha}_1 := t_1 : \mathbb{N} \dashv \Delta}{\Gamma_0[\hat{\alpha} : \mathbb{N}] \vdash \hat{\alpha} := \mathsf{succ}(t_1) : \mathbb{N} \dashv \Delta}\ \mathsf{InstSucc}$$

$$\begin{aligned}
|\mathsf{unsolved}(\Delta)| + 1 &= |\mathsf{unsolved}(\Gamma_0[\hat{\alpha}_1 : \mathbb{N}, \hat{\alpha} : \mathbb{N} = \mathsf{succ}(\hat{\alpha}_1)])| && \text{By i.h.}\\
\text{☞} \qquad &= |\mathsf{unsolved}(\Gamma_0[\hat{\alpha} : \mathbb{N}])| && \text{By definition of } \mathsf{unsolved}(-)
\end{aligned}$$
□

Lemma 75 (Checkeq Solving)**.** *If* $\Gamma \vdash s \mathrel{\hat{=}} t : \kappa \dashv \Delta$ *then either* $\Delta = \Gamma$ *or* $|\mathsf{unsolved}(\Delta)| < |\mathsf{unsolved}(\Gamma)|$.

Proof. By induction on the given derivation.

- **Case**
$$\frac{}{\Gamma \vdash u \mathrel{\hat{=}} u : \kappa \dashv \underbrace{\Gamma}_{\Delta}}\ \mathsf{CheckeqVar}$$
Here $\Delta = \Gamma$.

- **Cases** CheckeqUnit, CheckeqZero: Similar to the CheckeqVar case.

- **Case**
$$\frac{\Gamma \vdash \sigma \mathrel{\hat{=}} t : \mathbb{N} \dashv \Delta}{\Gamma \vdash \mathsf{succ}(\sigma) \mathrel{\hat{=}} \mathsf{succ}(t) : \mathbb{N} \dashv \Delta}\ \mathsf{CheckeqSucc}$$
Follows by i.h.

- **Case** $\dfrac{\Gamma_0[\hat{\alpha}] \vdash \hat{\alpha} := t : \kappa \dashv \Delta \qquad \hat{\alpha} \notin FV(t)}{\underbrace{\Gamma_0[\hat{\alpha}]}_{\Gamma} \vdash \hat{\alpha} \doteq t : \kappa \dashv \Delta}\ \mathsf{CheckeqInstL}$

  $\Gamma_0[\hat{\alpha}] \vdash \hat{\alpha} := t : \kappa \dashv \Delta$ Subderivation
  $\Gamma \vdash \hat{\alpha} := t : \kappa \dashv \Delta$ $\Gamma = \Gamma_0[\hat{\alpha}]$
  $\Delta = \Gamma$ or $|\mathsf{unsolved}(\Delta)| = |\mathsf{unsolved}(\Gamma)| - 1$ By Lemma 74 (Instantiation Solves)
  ☞ $\Delta = \Gamma$ or $|\mathsf{unsolved}(\Delta)| < |\mathsf{unsolved}(\Gamma)|$

- **Case** $\dfrac{\Gamma[\hat{\alpha} : \kappa] \vdash \hat{\alpha} := t : \kappa \dashv \Delta \qquad \hat{\alpha} \notin FV(t)}{\Gamma[\hat{\alpha} : \kappa] \vdash t \doteq \hat{\alpha} : \kappa \dashv \Delta}\ \mathsf{CheckeqInstR}$

  Similar to the CheckeqInstL case.

- **Case** $\dfrac{\Gamma \vdash \sigma_1 \doteq \tau_1 : \star \dashv \Theta \qquad \Theta \vdash [\Theta]\sigma_2 \doteq [\Theta]\tau_2 : \star \dashv \Delta}{\Gamma \vdash \underbrace{\sigma_1 \oplus \sigma_2}_{\sigma} \doteq \underbrace{\tau_1 \oplus \tau_2}_{t} : \star \dashv \Delta}\ \mathsf{CheckeqBin}$

  $\Gamma \vdash \sigma_1 \doteq \tau_1 : \star \dashv \Theta$ Subderivation
  $\Theta = \Gamma$ or $|\mathsf{unsolved}(\Theta)| < |\mathsf{unsolved}(\Gamma)|$ By i.h.

  – $\Theta = \Gamma$:
    $\Theta \vdash [\Theta]\sigma_2 \doteq [\Theta]\tau_2 : \star \dashv \Delta$ Subderivation
    $\Gamma \vdash [\Gamma]\sigma_2 \doteq [\Gamma]\tau_2 : \star \dashv \Delta$ By $\Theta = \Gamma$
    ☞ $\Delta = \Gamma$ or $|\mathsf{unsolved}(\Gamma)| = |\mathsf{unsolved}(\Delta)| + 1$ By i.h.

  – $|\mathsf{unsolved}(\Theta)| < |\mathsf{unsolved}(\Gamma)|$:
    $\Theta \vdash [\Theta]\sigma_2 \doteq [\Theta]\tau_2 : \star \dashv \Delta$ Subderivation
    $\Delta = \Theta$ or $|\mathsf{unsolved}(\Delta)| < |\mathsf{unsolved}(\Theta)|$ By i.h.
    If $\Delta = \Theta$ then substituting Δ for Θ in $|\mathsf{unsolved}(\Theta)| < |\mathsf{unsolved}(\Gamma)|$ gives $|\mathsf{unsolved}(\Delta)| < |\mathsf{unsolved}(\Gamma)|$.
    If $|\mathsf{unsolved}(\Delta)| < |\mathsf{unsolved}(\Theta)|$ then transitivity of $<$ gives $|\mathsf{unsolved}(\Delta)| < |\mathsf{unsolved}(\Gamma)|$. □

Lemma 76 (Prop Equiv Solving)**.**
If $\Gamma \vdash P \equiv Q \dashv \Delta$ *then either* $\Delta = \Gamma$ *or* $|\mathsf{unsolved}(\Delta)| < |\mathsf{unsolved}(\Gamma)|$.

Proof. Only one rule can derive the judgment:

- **Case** $\dfrac{\Gamma \vdash \sigma_1 \doteq t_1 : \mathbb{N} \dashv \Theta \qquad \Theta \vdash [\Theta]\sigma 2 \doteq [\Theta]t_2 : \mathbb{N} \dashv \Delta}{\Gamma \vdash (\sigma_1 = \sigma 2) \equiv (t_1 = t_2) \dashv \Delta}\ \equiv\mathsf{PropEq}$

  By Lemma 75 (Checkeq Solving) on the first premise,
  either $\Theta = \Gamma$ or $|\mathsf{unsolved}(\Theta)| < |\mathsf{unsolved}(\Gamma)|$.

  In the former case, the result follows from Lemma 75 (Checkeq Solving) on the second premise.

  In the latter case, applying Lemma 75 (Checkeq Solving) to the second premise either gives $\Delta = \Theta$, and therefore
  $$|\mathsf{unsolved}(\Delta)| < |\mathsf{unsolved}(\Gamma)|$$
  or gives $|\mathsf{unsolved}(\Delta)| < |\mathsf{unsolved}(\Theta)|$, which also leads to $|\mathsf{unsolved}(\Delta)| < |\mathsf{unsolved}(\Gamma)|$. □

Lemma 77 (Equiv Solving)**.**
If $\Gamma \vdash A \equiv B \dashv \Delta$ *then either* $\Delta = \Gamma$ *or* $|\mathsf{unsolved}(\Delta)| < |\mathsf{unsolved}(\Gamma)|$.

Proof. By induction on the given derivation.

- **Case**
  $$\frac{}{\Gamma \vdash \alpha \equiv \alpha \dashv \Gamma}\ \equiv\mathsf{Var}$$
  Here $\Delta = \Gamma$.

- **Cases** ≡Exvar, ≡Unit: Similar to the ≡Var case.

- **Case** $\dfrac{\Gamma \vdash A_1 \equiv B_1 \dashv \Theta \qquad \Theta \vdash [\Theta]A_2 \equiv [\Theta]B_2 \dashv \Delta}{\Gamma \vdash (A_1 \oplus A_2) \equiv (B_1 \oplus B_2) \dashv \Delta} \equiv\oplus$

  By i.h., either $\Theta = \Gamma$ or $|\mathsf{unsolved}(\Theta)| < |\mathsf{unsolved}(\Gamma)|$.

  In the former case, apply the i.h. to the second premise. Now either $\Delta = \Theta$—and therefore $\Delta = \Gamma$—or $|\mathsf{unsolved}(\Delta)| < |\mathsf{unsolved}(\Theta)|$. Since $\Theta = \Gamma$, we have $|\mathsf{unsolved}(\Delta)| < |\mathsf{unsolved}(\Gamma)|$.

  In the latter case, we have $|\mathsf{unsolved}(\Theta)| < |\mathsf{unsolved}(\Gamma)|$. By i.h. on the second premise, either $\Delta = \Theta$, and substituting Δ for Θ gives $|\mathsf{unsolved}(\Delta)| < |\mathsf{unsolved}(\Gamma)|$—or $|\mathsf{unsolved}(\Delta)| < |\mathsf{unsolved}(\Theta)|$, which combined with $|\mathsf{unsolved}(\Theta)| < |\mathsf{unsolved}(\Gamma)|$ gives $|\mathsf{unsolved}(\Delta)| < |\mathsf{unsolved}(\Gamma)|$.

- **Case** ≡Vec: Similar to the $\equiv\oplus$ case.

- **Case** $\dfrac{\Gamma, \alpha : \kappa \vdash A_0 \equiv B_0 \dashv \Delta, \alpha : \kappa, \Delta'}{\Gamma \vdash \forall \alpha : \kappa.\, A_0 \equiv \forall \alpha : \kappa.\, B_0 \dashv \Delta} \equiv\forall$

  By i.h., either $(\Delta, \alpha : \kappa, \Delta') = (\Gamma, \alpha : \kappa)$, or $|\mathsf{unsolved}(\Delta, \alpha : \kappa, \Delta')| < |\mathsf{unsolved}(\Gamma, \alpha : \kappa)|$.

  In the former case, Lemma 22 (Extension Inversion) (i) tells us that $\Delta' = \cdot$. Thus, $(\Delta, \alpha : \kappa) = (\Gamma, \alpha : \kappa)$, and so $\Delta = \Gamma$.

  In the latter case, we have $|\mathsf{unsolved}(\Delta, \alpha : \kappa, \Delta')| < |\mathsf{unsolved}(\Gamma, \alpha : \kappa)|$, that is:

  $$|\mathsf{unsolved}(\Delta)| + 0 + |\mathsf{unsolved}(\Delta')| \quad < \quad |\mathsf{unsolved}(\Gamma)| + 0$$

  Since $|\mathsf{unsolved}(\Delta')|$ cannot be negative, we have $|\mathsf{unsolved}(\Delta)| < |\mathsf{unsolved}(\Gamma)|$.

- **Case** $\dfrac{\Gamma \vdash P \equiv Q \dashv \Theta \qquad \Theta \vdash [\Theta]A_0 \equiv [\Theta]B_0 \dashv \Delta}{\Gamma \vdash P \supset A_0 \equiv Q \supset B_0 \dashv \Delta} \equiv\supset$

  Similar to the $\equiv\oplus$ case, but using Lemma 76 (Prop Equiv Solving) on the first premise instead of the i.h.

- **Case** $\dfrac{\Gamma \vdash P \equiv Q \dashv \Theta \qquad \Theta \vdash [\Theta]A_0 \equiv [\Theta]B_0 \dashv \Delta}{\Gamma \vdash A_0 \wedge P \equiv B_0 \wedge Q \dashv \Delta} \equiv\wedge$

  Similar to the $\equiv\wedge$ case.

- **Case** $\dfrac{\Gamma_0[\hat\alpha] \vdash \hat\alpha := \tau : \star \dashv \Delta \qquad \hat\alpha \notin \mathrm{FV}(\tau)}{\underbrace{\Gamma_0[\hat\alpha]}_{\Gamma} \vdash \hat\alpha \equiv \tau \dashv \Delta} \equiv\mathsf{InstantiateL}$

  By Lemma 74 (Instantiation Solves), $|\mathsf{unsolved}(\Delta)| = |\mathsf{unsolved}(\Gamma)| - 1$.

- **Case** $\dfrac{\Gamma_0[\hat\alpha] \vdash \hat\alpha := \tau : \star \dashv \Delta \qquad \hat\alpha \notin \mathrm{FV}(\tau)}{\Gamma_0[\hat\alpha] \vdash \tau \equiv \hat\alpha \dashv \Delta} \equiv\mathsf{InstantiateR}$

  Similar to the ≡InstantiateL case. □

Lemma 78 (Decidability of Propositional Judgments)**.**
The following judgments are decidable, with Δ as output in (1)–(3), and $\Delta^\perp$ as output in (4) and (5).

We assume $\sigma = [\Gamma]\sigma$ and $t = [\Gamma]t$ in (1) and (4). Similarly, in the other parts we assume $P = [\Gamma]P$ and (in part (3)) $Q = [\Gamma]Q$.

(1) $\Gamma \vdash \sigma \doteq t : \kappa \dashv \Delta$

(2) $\Gamma \vdash P\ \mathit{true} \dashv \Delta$

(3) $\Gamma \vdash P \equiv Q \dashv \Delta$

(4) $\Gamma \mathbin{/} \sigma \mathrel{\hat{=}} t : \kappa \dashv \Delta^{\perp}$

(5) $\Gamma \mathbin{/} P \dashv \Delta^{\perp}$

Proof. Since there is no mutual recursion between the judgments, we can prove their decidability in order, separately.

(1) *Decidability of* $\Gamma \vdash \sigma \mathrel{\hat{=}} t : \kappa \dashv \Delta$: By induction on the sizes of σ and t.

- **Cases** CheckeqVar, CheckeqUnit, CheckeqZero: No premises.
- **Case** CheckeqSucc: Both σ and t get smaller in the premise.
- **Cases** CheckeqInstL, CheckeqInstR: Follows from Lemma 67 (Decidability of Instantiation).

(2) *Decidability of* $\Gamma \vdash P$ *true* $\dashv \Delta$: By induction on σ and t. But we have only one rule deriving this judgment form, CheckpropEq, which has the judgment in (1) as a premise, so decidability follows from part (1).

(3) *Decidability of* $\Gamma \vdash P \equiv Q \dashv \Delta$: By induction on P and Q. But we have only one rule deriving this judgment form, $\equiv$PropEq, which has two premises of the form (1), so decidability follows from part (1).

(4) *Decidability of* $\Gamma \mathbin{/} \sigma \mathrel{\hat{=}} t : \kappa \dashv \Delta^{\perp}$: By lexicographic induction, first on the number of unsolved variables (both universal and existential) in Γ, then on σ and t. We also show that the number of unsolved variables is nonincreasing in the output context (if it exists).

- **Cases** ElimeqUvarRefl, ElimeqZero: No premises, and the output is the same as the input.
- **Case** ElimeqClash: The only premise is the clash judgment, which is clearly decidable. There is no output.
- **Case** ElimeqBin: In the first premise, we have the same Γ but both σ and t are smaller. By i.h., the first premise is decidable; moreover, either some variables in Θ were solved, or no additional variables were solved.

  If some variables in Θ were solved, the second premise is smaller than the conclusion according to our lexicographic measure, so by i.h., the second premise is decidable.

  If no additional variables were solved, then $\Theta = \Gamma$. Therefore $[\Theta]\tau_2 = [\Gamma]\tau_2$. It is given that $\sigma = [\Gamma]\sigma$ and $t = [\Gamma]t$, so $[\Gamma]\tau_2 = \tau_2$. Likewise, $[\Theta]\tau_2' = [\Gamma]\tau_2' = \tau_2'$, so we aremaking a recursive call on a strictly smaller subterm.

  Regardless, $\Delta^{\perp}$ is either $\perp$, or is a Δ which has no more unsolved variables than Θ, which in turn has no more unsolved variables than Γ.
- **Case** ElimeqBinBot:

  The premise is invoked on subterms, and does not yield an output context.
- **Case** ElimeqSucc: Both σ and t get smaller. By i.h., the output context has fewer unsolved variables, if it exists.
- **Cases** ElimeqInstL, ElimeqInstR: Follows from Lemma 67 (Decidability of Instantiation). Furthermore, by Lemma 74 (Instantiation Solves), instantiation solves a variable in the output.
- **Cases** ElimeqUvarL, ElimeqUvarR: These rules have no nontrivial premises, and α is solved in the output context.
- **Cases** ElimeqUvarL$\perp$, ElimeqUvarR$\perp$: These rules have no nontrivial premises, and produce the output context $\perp$.

(5) *Decidability of* $\Gamma \mathbin{/} P \dashv \Delta^{\perp}$: By induction on P. But we have only one rule deriving this judgment form, ElimpropEq, for which decidability follows from part (4). □

Lemma 79 (Decidability of Equivalence)**.**
Given a context Γ *and types* A, B *such that* $\Gamma \vdash A$ *type and* $\Gamma \vdash B$ *type and* $[\Gamma]A = A$ *and* $[\Gamma]B = B$*, it is decidable whether there exists* Δ *such that* $\Gamma \vdash A \equiv B \dashv \Delta$*.*

Proof. Let the judgment $\Gamma \vdash A \equiv B \dashv \Delta$ be measured lexicographically by

(E1) $\#\mathsf{large}(A) + \#\mathsf{large}(B)$;

(E2) $|\mathsf{unsolved}(\Gamma)|$, the number of unsolved existential variables in Γ;

(E3) $|A| + |B|$.

- **Cases** $\equiv$Var, $\equiv$Exvar, $\equiv$Unit: No premises.
- **Case** $\dfrac{\Gamma \vdash A_1 \equiv B_1 \dashv \Theta \qquad \Theta \vdash [\Theta]A_2 \equiv [\Theta]B_2 \dashv \Delta}{\Gamma \vdash A_1 \oplus A_2 \equiv B_1 \oplus B_2 \dashv \Delta} \equiv\oplus$

  In the first premise, part (E1) either gets smaller (if A_2 or B_2 have large connectives) or stays the same. Since the first premise has the same input context, part (E2) remains the same. However, part (E3) gets smaller.

  In the second premise, part (E1) either gets smaller (if A_1 or B_1 have large connectives) or stays the same.
- **Case** $\equiv$Vec: Similar to a special case of $\equiv\oplus$, where two of the types are monotypes.
- **Case** $\dfrac{\Gamma, \alpha : \kappa \vdash A_0 \equiv B_0 \dashv \Delta, \alpha : \kappa, \Delta'}{\Gamma \vdash \underbrace{\forall \alpha : \kappa.\, A_0}_{A} \equiv \underbrace{\forall \alpha : \kappa.\, B_0}_{B} \dashv \Delta} \equiv\forall$

  Since $\#\mathsf{large}(A_0) + \#\mathsf{large}(B_0) = \#\mathsf{large}(A) + \#\mathsf{large}(B) - 2$, the first part of the measure gets smaller.
- **Case** $\dfrac{\Gamma \vdash P \equiv Q \dashv \Theta \qquad \Theta \vdash [\Theta]A_0 \equiv [\Theta]B_0 \dashv \Delta}{\Gamma \vdash \underbrace{P \supset A_0}_{A} \equiv \underbrace{Q \supset B_0}_{B} \dashv \Delta} \equiv\supset$

  The first premise is decidable by Lemma 78 (Decidability of Propositional Judgments) (3).

  For the second premise, by Lemma 73 (Substitution Isn't Large), $\#\mathsf{large}([\Theta]A_0) = \#\mathsf{large}(A_0)$ and $\#\mathsf{large}([\Theta]B_0) = \#\mathsf{large}(B_0)$. Since $\#\mathsf{large}(A) = \#\mathsf{large}(A_0) + 1$ and $\#\mathsf{large}(B) = \#\mathsf{large}(B_0) + 1$, we have
  $$\#\mathsf{large}([\Theta]A_0) + \#\mathsf{large}([\Theta]B_0) \;<\; \#\mathsf{large}(A) + \#\mathsf{large}(B)$$
  which makes the first part of the measure smaller.
- **Case** $\dfrac{\Gamma \vdash P \equiv Q \dashv \Theta \qquad \Theta \vdash [\Theta]A_0 \equiv [\Theta]B_0 \dashv \Delta}{\Gamma \vdash A_0 \wedge P \equiv B_0 \wedge Q \dashv \Delta} \equiv\wedge$

  Similar to the $\equiv\supset$ case.
- **Case** $\dfrac{\Gamma[\hat{\alpha}] \vdash \hat{\alpha} := \tau : \star \dashv \Delta \qquad \hat{\alpha} \notin \mathrm{FV}(\tau)}{\Gamma[\hat{\alpha}] \vdash \hat{\alpha} \equiv \tau \dashv \Delta} \equiv\mathsf{InstantiateL}$

  Follows from Lemma 67 (Decidability of Instantiation).
- **Case** $\equiv$InstantiateR: Similar to the $\equiv$InstantiateL case. □

I′.2 Decidability of Subtyping

Theorem 1 (Decidability of Subtyping)**.**
Given a context Γ *and types* A, B *such that* $\Gamma \vdash A$ *type and* $\Gamma \vdash B$ *type and* $[\Gamma]A = A$ *and* $[\Gamma]B = B$*, it is decidable whether there exists* Δ *such that* $\Gamma \vdash A <:^{\pm} B \dashv \Delta$.

Proof. Let the judgments be measured lexicographically by $\#\mathsf{large}(A) + \#\mathsf{large}(B)$.

For each subtyping rule, we show that every premise is smaller than the conclusion, or already known to be decidable. The condition that $[\Gamma]A = A$ and $[\Gamma]B = B$ is easily satisfied at each inductive step, using the definition of substitution.

Now, we consider the rules deriving $\Gamma \vdash A <:^{\pm} B \dashv \Delta$.

- **Case**
$$\frac{\begin{array}{l}A \text{ not headed by } \forall/\exists \\ B \text{ not headed by } \forall/\exists \qquad \Gamma \vdash A \equiv B \dashv \Delta\end{array}}{\Gamma \vdash A <:^{\pm} B \dashv \Delta} <:\mathsf{Equiv}$$
In this case, we appeal to Lemma 79 (Decidability of Equivalence).

- **Case**
$$\frac{\begin{array}{l}B \text{ not headed by } \forall \\ \Gamma, \blacktriangleright_{\hat{\alpha}}, \hat{\alpha} : \kappa \vdash [\hat{\alpha}/\alpha]A <:^{-} B \dashv \Delta, \blacktriangleright_{\hat{\alpha}}, \Theta\end{array}}{\Gamma \vdash \forall \alpha : \kappa.\, A <:^{-} B \dashv \Delta} <:\forall\mathsf{L}$$
The premise has one fewer quantifier.

- **Case**
$$\frac{\Gamma, \beta : \kappa \vdash A <:^{-} B \dashv \Delta, \beta : \kappa, \Theta}{\Gamma \vdash A <:^{-} \forall \beta : \kappa.\, B \dashv \Delta} <:\forall\mathsf{R}$$
The premise has one fewer quantifier.

- **Case**
$$\frac{\Gamma, \alpha : \kappa \vdash A <:^{+} B \dashv \Delta, \alpha : \kappa, \Theta}{\Gamma \vdash \exists \alpha : \kappa.\, A <:^{+} B \dashv \Delta} <:\exists\mathsf{L}$$
The premise has one fewer quantifier.

- **Case**
$$\frac{\begin{array}{l}A \text{ not headed by } \exists \\ \Gamma, \blacktriangleright_{\hat{\beta}}, \hat{\beta} : \kappa \vdash A <:^{+} [\hat{\beta}/\beta]B \dashv \Delta, \blacktriangleright_{\hat{\beta}}, \Theta\end{array}}{\Gamma \vdash A <:^{+} \exists \beta : \kappa.\, B \dashv \Delta} <:\exists\mathsf{R}$$
The premise has one fewer quantifier.

- **Case**
$$\frac{\Gamma \vdash A <:^{-} B \dashv \Delta \qquad \begin{array}{l}neg(A) \\ nonpos(B)\end{array}}{\Gamma \vdash A <:^{+} B \dashv \Delta} <:^{-}_{+}\mathsf{L}$$
Consider whether B is negative.
  - Case $neg(B)$:
$$\begin{array}{rl} B = \forall \beta : \kappa.\, B' & \text{Definition of } neg(B) \\ \Gamma, \beta : \kappa \vdash A <:^{-} B' \dashv \Delta, \beta : \kappa, \Theta & \text{Inversion on the premise}\end{array}$$
There is one fewer quantifier in the subderivation.
  - Case $nonneg(B)$:
In this case, B is not headed by a $\forall$.
$$\begin{array}{rl} A = \forall \alpha : \kappa.\, A' & \text{Definition of } neg(A) \\ \Gamma, \blacktriangleright_{\hat{\alpha}}, \hat{\alpha} : \kappa \vdash [\hat{\alpha}/\alpha]A' <:^{-} ' \dashv \Delta, \blacktriangleright_{\hat{\alpha}}, \Theta & \text{Inversion on the premise}\end{array}$$
There is one fewer quantifier in the subderivation.

- **Case**
$$\frac{\Gamma \vdash A <:^{-} B \dashv \Delta \qquad \begin{array}{l}nonpos(A) \\ neg(B)\end{array}}{\Gamma \vdash A <:^{+} B \dashv \Delta} <:^{-}_{+}\mathsf{R}$$
$$\begin{array}{rl} B = \forall \beta : \kappa.\, B' & \text{Definition of } neg(B) \\ \Gamma, \beta : \kappa \vdash A <:^{-} B' \dashv \Delta, \beta : \kappa, \Theta & \text{Inversion on the premise}\end{array}$$
There is one fewer quantifier in the subderivation.

- **Case**
$$\frac{\Gamma \vdash A <:^{+} B \dashv \Delta \qquad \begin{array}{l}pos(A) \\ nonneg(B)\end{array}}{\Gamma \vdash A <:^{-} B \dashv \Delta} <:^{+}_{-}\mathsf{L}$$
This case is similar to the $<:^{-}_{+}\mathsf{R}$ case.

- **Case**
$$\frac{\Gamma \vdash A <:^{+} B \dashv \Delta \qquad \begin{array}{l} nonneg(A) \\ pos(B) \end{array}}{\Gamma \vdash A <:^{-} B \dashv \Delta} <:^{+}_{-}R$$
This case is similar to the $<:^{-}_{+}L$ case.

□

I′.3 Decidability of Matching and Coverage

Lemma 80 (Decidability of Expansion Judgments)**.**
Given branches Π*, it is decidable whether:*

(1) there exists Π' *such that* $\Pi \overset{\times}{\rightsquigarrow} \Pi'$*;*

(2) there exist Π_L *and* Π_R *such that* $\Pi \overset{+}{\rightsquigarrow} \Pi_L \parallel \Pi_R$*;*

(3) there exists Π' *such that* $\Pi \overset{\text{var}}{\rightsquigarrow} \Pi'$*;*

(4) there exists Π' *such that* $\Pi \overset{1}{\rightsquigarrow} \Pi'$*.*

Proof. In each part, by induction on Π: Every rule either has no premises, or breaks down Π in its nontrivial premise. □

Theorem 2 (Decidability of Coverage)**.**
Given a context Γ*, branches* Π *and types* $\vec{A}$*, it is decidable whether* $\Gamma \vdash \Pi$ *covers* $\vec{A}$ *is derivable.*

Proof. By induction on, lexicographically, (1) the number of $\wedge$ connectives appearing in $\vec{A}$, and then (2) the size of $\vec{A}$, considered to be the sum of the sizes $|A|$ of each type A in $\vec{A}$.

(For CoversVar, Covers$\times$, and Covers$+$, we also use the appropriate part of Lemma 80 (Decidability of Expansion Judgments).)

- **Case** CoversEmpty: No premises.
- **Case** CoversVar: The number of $\wedge$ connectives does not grow, and $\vec{A}$ gets smaller.
- **Case** Covers1: The number of $\wedge$ connectives does not grow, and $\vec{A}$ gets smaller.
- **Case** Covers$\times$: The number of $\wedge$ connectives does not grow, and $\vec{A}$ gets smaller, since $|A_1|+|A_2| < |A_1 \times A_2|$.
- **Case** Covers$+$: Here we have $\vec{A} = (A_1 + A_2, \vec{B})$. In the first premise, we have $(A_1, \vec{B})$, which is smaller than $\vec{A}$, and in the second premise we have $(A_2, \vec{B})$, which is likewise smaller. (In both premises, the number of $\wedge$ connectives does not grow.)
- **Case** Covers$\exists$: The number of $\wedge$ connectives does not grow, and $\vec{A}$ gets smaller.
- **Case** CoversEq: The first premise is decidable by Lemma 78 (Decidability of Propositional Judgments) (4). The number of $\wedge$ connectives in $\vec{A}$ gets smaller (note that applying Δ as a substitution cannot add $\wedge$ connectives).
- **Case** CoversEqBot: Decidable by Lemma 78 (Decidability of Propositional Judgments) (4). □

I′.4 Decidability of Typing

Theorem 3 (Decidability of Typing)**.**

(i) Synthesis: *Given a context* Γ*, a principality* p*, and a term* e*,*
it is decidable whether there exist a type A *and a context* Δ *such that*
$\Gamma \vdash e \Rightarrow A\ p \dashv \Delta$.

(ii) Spines: *Given a context* Γ*, a spine* s*, a principality* p*, and a type* A *such that* $\Gamma \vdash A$ *type,*
it is decidable whether there exist a type B*, a principality* q *and a context* Δ *such that*
$\Gamma \vdash s : A\ p \gg B\ q \dashv \Delta$.

(iii) Checking: *Given a context* Γ*, a principality* p*, a term* e*, and a type* B *such that* $\Gamma \vdash \mathrm{B}$ *type, it is decidable whether there is a context* Δ *such that* $\Gamma \vdash e \Leftarrow \mathrm{B}\ \mathrm{p} \dashv \Delta$.

(iv) Matching: *Given a context* Γ*, branches* Π*, a list of types* $\vec{A}$*, a type* C*, and a principality* p*, it is decidable whether there exists* Δ *such that* $\Gamma \vdash \Pi :: \vec{A} \Leftarrow \mathrm{C}\ \mathrm{p} \dashv \Delta$.

Also, if given a proposition P *as well, it is decidable whether there exists* Δ *such that* $\Gamma \ / \ \mathrm{P} \vdash \Pi :: \vec{A} \Leftarrow \mathrm{C}\ \mathrm{p} \dashv \Delta$.

Proof. For rules deriving judgments of the form

$$\begin{array}{l}\Gamma \vdash e \Rightarrow - - \dashv -\\ \Gamma \vdash e \Leftarrow \mathrm{B}\ \mathrm{p} \dashv -\\ \Gamma \vdash s : \mathrm{B}\ \mathrm{p} \gg - - \dashv -\\ \Gamma \vdash \Pi :: \vec{A} \Leftarrow \mathrm{C}\ \mathrm{p} \dashv -\end{array}$$

(where we write "–" for parts of the judgments that are outputs), the following induction measure on such judgments is adequate to prove decidability:

$$\left\langle e/s/\Pi, \quad \begin{array}{llll} \Rightarrow & & & \\ \Leftarrow / \gg, & \#\mathsf{large}(\mathrm{B}), & \mathrm{B} & \\ \mathsf{Match}, & \vec{A}, & \text{match judgment form} & \end{array} \right\rangle$$

where $\langle \ldots \rangle$ denotes lexicographic order, and where (when comparing two judgments typing terms of the same size) the synthesis judgment (top line) is considered smaller than the checking judgment (second line). That is,

$$\Rightarrow \quad \prec \quad \Leftarrow / \gg / \mathsf{Match}$$

Two match judgments are compared according to, first, the list of branches Π (which is a subterm of the containing case expression, allowing us to invoke the i.h. for the Case rule), then the size of the list of types $\vec{A}$ (considered to be the sum of the sizes $|A|$ of each type A in $\vec{A}$), and then, finally, whether the judgment is $\Gamma/\mathrm{P} \vdash \ldots$ or $\Gamma \vdash \ldots$, considering the former judgment ($\Gamma/\mathrm{P} \vdash \ldots$) to be larger.

Note that this measure only uses the input parts of the judgments, leading to a straightforward decidability argument.

We will show that in each rule deriving a synthesis, checking, spine or match judgment, every premise is smaller than the conclusion.

- **Case** EmptySpine: No premises.
- **Case** $\rightarrow$Spine: In each premise, the expression/spine gets smaller (we have $e\ s$ in the conclusion, e in the first premise, and s in the second premise).
- **Case** Var: No nontrivial premises.
- **Case** Sub: The first premise has the same subject term e as the conclusion, but the judgment is smaller because our measure considers synthesis to be smaller than checking.

  The second premise is a subtyping judgment, which by Theorem 1 (Decidability of Subtyping) is decidable.
- **Case** Anno: It is easy to show that the judgment $\Gamma \vdash A\ \mathbf{!}\ \mathit{type}$ is decidable. The second premise types e, but the conclusion types $(e : A)$, so the first part of the measure gets smaller.
- **Cases** 1I, 1I$\hat{\alpha}$: No premises.
- **Case** $\forall$I: Both the premise and conclusion type e, and both are checking; however, $\#\mathsf{large}(A_0) < \#\mathsf{large}(\forall \alpha : \kappa.\ A_0)$, so the premise is smaller.
- **Case** $\forall$Spine: Both the premise and conclusion type $e\ s$, and both are spine judgments; however, $\#\mathsf{large}(-)$ decreases.
- **Case** $\wedge$I: By Lemma 78 (Decidability of Propositional Judgments) (2), the first premise is decidable. For the second premise, $\#\mathsf{large}([\Theta]A_0) = \#\mathsf{large}(A_0) < \#\mathsf{large}(A_0 \wedge \mathrm{P})$.

- **Case** $\supset$I: For the first premise, use Lemma 78 (Decidability of Propositional Judgments) (5). In the second premise, #large$(-)$ gets smaller (similar to the $\wedge$I case).
- **Case** $\supset$I$\bot$: The premise is decidable by Lemma 78 (Decidability of Propositional Judgments) (5).
- **Case** $\supset$Spine: Similar to the $\wedge$I case.
- **Cases** $\to$I, $\to$I$\hat{\alpha}$: In the premise, the term is smaller.
- **Cases** $\to$E, $\to$E-!: In all premises, the term is smaller.
- **Cases** $+\mathsf{I}_k$, $+\mathsf{I}\hat{\alpha}_k$, $\times$I, $\times$I$\hat{\alpha}$: In all premises, the term is smaller.
- **Case** Case: In the first premise, the term is smaller. In the second premise, we have a list of branches that is a proper subterm of the case expression. The third premise is decidable by Theorem 2 (Decidability of Coverage).

We now consider the match rules:

- **Case** MatchEmpty: No premises.
- **Case** MatchSeq: In each premise, the list of branches is properly contained in Π, making each premise smaller by the first part ("$e/s/\Pi$") of the measure.
- **Case** MatchBase: The term e in the premise is properly contained in Π.
- **Cases** Match$\exists$, Match$\times$, Match$+_k$, MatchNeg, MatchWild: Smaller by part (2) of the measure.
- **Case** Match$\wedge$: The premise has a smaller $\vec{A}$, so it is smaller by the $\vec{A}$ part of the measure. (The premise is the other judgment form, so it is *larger* by the "match judgment form" part, but $\vec{A}$ lexicographically dominates.)
- **Case** Match$\bot$: For the premise, use Lemma 78 (Decidability of Propositional Judgments) (4).
- **Case** MatchUnify:

  Lemma 78 (Decidability of Propositional Judgments) (4) shows that the first premise is decidable. The second premise has the same (single) branch and list of types, but is smaller by the "match judgment form" part of the measure. □

J′ Determinacy

Lemma 81 (Determinacy of Auxiliary Judgments)**.**

(1) Elimeq: *Given* Γ, σ, t, κ *such that* $\mathsf{FEV}(\sigma) \cup \mathsf{FEV}(t) = \emptyset$ *and* $\mathcal{D}_1 :: \Gamma \;/\; \sigma \triangleq t : \kappa \dashv \Delta_1^\bot$ *and* $\mathcal{D}_2 :: \Gamma \;/\; \sigma \triangleq t : \kappa \dashv \Delta_2^\bot$,
it is the case that $\Delta_1^\bot = \Delta_2^\bot$.

(2) Instantiation: *Given* Γ, $\hat{\alpha}$, t, κ *such that* $\hat{\alpha} \in \mathsf{unsolved}(\Gamma)$ *and* $\Gamma \vdash t : \kappa$ *and* $\hat{\alpha} \notin \mathsf{FV}(t)$
and $\mathcal{D}_1 :: \Gamma \vdash \hat{\alpha} := t : \kappa \dashv \Delta_1$ *and* $\mathcal{D}_2 :: \Gamma \vdash \hat{\alpha} := t : \kappa \dashv \Delta_2$
it is the case that $\Delta_1 = \Delta_2$.

(3) Symmetric instantiation:

Given Γ, $\hat{\alpha}$, $\hat{\beta}$, κ *such that* $\hat{\alpha}, \hat{\beta} \in \mathsf{unsolved}(\Gamma)$ *and* $\hat{\alpha} \neq \hat{\beta}$
and $\mathcal{D}_1 :: \Gamma \vdash \hat{\alpha} := \hat{\beta} : \kappa \dashv \Delta_1$ *and* $\mathcal{D}_2 :: \Gamma \vdash \hat{\beta} := \hat{\alpha} : \kappa \dashv \Delta_2$
it is the case that $\Delta_1 = \Delta_2$.

(4) Checkeq: *Given* Γ, σ, t, κ *such that* $\mathcal{D}_1 :: \Gamma \vdash \sigma \triangleq t : \kappa \dashv \Delta_1$ *and* $\mathcal{D}_2 :: \Gamma \vdash \sigma \triangleq t : \kappa \dashv \Delta_2$
it is the case that $\Delta_1 = \Delta_2$.

(5) Elimprop: *Given* Γ, P *such that* $\mathcal{D}_1 :: \Gamma \;/\; P \dashv \Delta_1^\bot$ *and* $\mathcal{D}_2 :: \Gamma \;/\; P \dashv \Delta_2^\bot$
it is the case that $\Delta_1 = \Delta_2$.

(6) Checkprop: *Given* Γ, P *such that* $\mathcal{D}_1 :: \Gamma \vdash P$ *true* $\dashv \Delta_1$ *and* $\mathcal{D}_2 :: \Gamma \vdash P$ *true* $\dashv \Delta_2$,
it is the case that $\Delta_1 = \Delta_2$.

Proof.
Proof of Part (1) (Elimeq).
Rule ElimeqZero applies if and only if $\sigma = t = \mathsf{zero}$.
Rule ElimeqSucc applies if and only if σ and t are headed by succ.
Now suppose $\sigma = \alpha$.

- Rule ElimeqUvarRefl applies if and only if $t = \alpha$. (Rule ElimeqClash cannot apply; rules ElimeqUvarL and ElimeqUvarR have a free variable condition; rules ElimeqUvarL$\perp$ and ElimeqUvarR$\perp$ have a condition that $\sigma \neq t$.)

  In the remainder, assume $t \neq \mathfrak{alpha}$.

- If $\alpha \in \mathrm{FV}(t)$, then rule ElimeqUvarL$\perp$ applies, and no other rule applies (including ElimeqUvarR$\perp$ and ElimeqClash).

  In the remainder, assume $\alpha \notin \mathrm{FV}(t)$.

- Consider whether ElimeqUvarR$\perp$ applies. The conclusion matches if we have $t = \beta$ for some $\beta \neq \alpha$ (that is, $\sigma = \alpha$ and $t = \beta$). But ElimeqUvarR$\perp$ has a condition that $\beta \in \mathrm{FV}(\sigma)$, and $\sigma = \alpha$, so the condition is not satisfied.

In the symmetric case, use the reasoning above, exchanging L's and R's in the rule names.

Proof of Part (2) (Instantiation).
Rule InstBin applies if and only if t has the form $t_1 \oplus t_2$.
Rule InstZero applies if and only if t has the form zero.
Rule InstSucc applies if and only if t has the form $\mathsf{succ}(t_0)$.
If t has the form $\hat{\beta}$, then consider whether $\hat{\beta}$ is declared to the left of $\hat{\alpha}$ in the given context:

- If $\hat{\beta}$ is declared to the left of $\hat{\alpha}$, then rule InstReach cannot be used, which leaves only InstSolve.
- If $\hat{\beta}$ is declared to the right of $\hat{\alpha}$, then InstSolve cannot be used because $\hat{\beta}$ is not well-formed under Γ_0 (the context to the left of $\hat{\alpha}$ in InstSolve). That leaves only InstReach.
- $\hat{\alpha}$ cannot be $\hat{\beta}$, because it is given that $\hat{\alpha} \notin \mathrm{FV}(t) = \mathrm{FV}(\hat{\beta}) = \{\hat{\beta}\}$.

Proof of Part (3) (Symmetric instantiation).
InstBin, InstZero and InstSucc cannot have been used in either derivation.
Suppose that InstSolve concluded $\mathcal{D}_1$. Then Δ_1 is the same as Γ with $\hat{\alpha}$ solved to $\hat{\beta}$. Moreover, $\hat{\beta}$ is declared to the left of $\hat{\alpha}$ in Γ. Thus, InstSolve cannot conclude $\mathcal{D}_2$. However, InstReach can conclude $\mathcal{D}_2$, but produces a context Δ_2 which is the same as Γ but with $\hat{\alpha}$ solved to $\hat{\beta}$. Therefore $\Delta_1 = \Delta_2$.
The other possibility is that InstReach concluded $\mathcal{D}_1$. Then Δ_1 is the same as Γ with $\hat{\beta}$ solved to $\hat{\alpha}$, with $\hat{\alpha}$ declared to the left of $\hat{\beta}$ in Γ. Thus, InstReach cannot conclude $\mathcal{D}_2$. However, InstSolve can conclude $\mathcal{D}_2$, producing a context Δ_2 which is the same as Γ but with $\hat{\beta}$ solved to $\hat{\alpha}$. Therefore $\Delta_1 = \Delta_2$.

Proof of Part (4) (Checkeq).
Rule CheckeqVar applies if and only if $\sigma = t = \hat{\alpha}$ or $\sigma = t = \alpha$ (note the free variable conditions in CheckeqInstL and CheckeqInstR).
Rule CheckeqUnit applies if and only if $\sigma = t = 1$.
Rule CheckeqBin applies if and only if σ and t are both headed by the same binary connective.
Rule CheckeqZero applies if and only if $\sigma = t = \mathsf{zero}$.
Rule CheckeqSucc applies if and only if σ and t are headed by succ.
Now suppose $\sigma = \hat{\alpha}$. If t is not an existential variable, then CheckeqInstR cannot be used, which leaves only CheckeqInstL. If t is an existential variable, that is, some $\hat{\beta}$ (distinct from $\hat{\alpha}$), and is unsolved, then both CheckeqInstL and CheckeqInstR apply, but by part (3), we get the same output context from each.
The $t = \hat{\alpha}$ subcase is similar.

Proof of Part (5) (Elimprop). There is only one rule deriving this judgment; the result follows by part (1).

Proof of Part (6) (Checkprop). There is only one rule deriving this judgment; the result follows by part (4). □

Lemma 82 (Determinacy of Equivalence)**.**

(1) Propositional equivalence: *Given* Γ, P, Q *such that* $\mathcal{D}_1 :: \Gamma \vdash P \equiv Q \dashv \Delta_1$ *and* $\mathcal{D}_2 :: \Gamma \vdash P \equiv Q \dashv \Delta_2$, *it is the case that* $\Delta_1 = \Delta_2$.

(2) Type equivalence: *Given* Γ, A, B *such that* $\mathcal{D}_1 :: \Gamma \vdash A \equiv B \dashv \Delta_1$ *and* $\mathcal{D}_2 :: \Gamma \vdash A \equiv B \dashv \Delta_2$, *it is the case that* $\Delta_1 = \Delta_2$.

Proof.
Proof of Part (1) (propositional equivalence). Only one rule derives judgments of this form; the result follows from Lemma 81 (Determinacy of Auxiliary Judgments) (4).

Proof of Part (2) (type equivalence). If neither A nor B is an existential variable, they must have the same head connectives, and the same rule must conclude both derivations.

If A and B are the same existential variable, then only ≡Exvar applies (due to the free variable conditions in ≡InstantiateL and ≡InstantiateR).

If A and B are different unsolved existential variables, the judgment matches the conclusion of both ≡InstantiateL and ≡InstantiateR, but by part (3) of Lemma 81 (Determinacy of Auxiliary Judgments), we get the same output context regardless of which rule we choose. □

Theorem 4 (Determinacy of Subtyping)**.**

(1) Subtyping: *Given* Γ, e, A, B *such that* $\mathcal{D}_1 :: \Gamma \vdash A <:^{\pm} B \dashv \Delta_1$ *and* $\mathcal{D}_2 :: \Gamma \vdash A <:^{\pm} B \dashv \Delta_2$, *it is the case that* $\Delta_1 = \Delta_2$.

Proof. First, we consider whether we are looking at positive or negative subtyping, and then consider the outermost connective of A and B:

- If $\Gamma \vdash A <:^{+} B \dashv \Delta_1$ and $\Gamma \vdash A <:^{+} B \dashv \Delta_2$, then we know the last rule ending the derivation of $\mathcal{D}_1$ and $\mathcal{D}_2$ must be:

		B: ∀	B: ∃	B: other
A	∀	$<:^{-}_{+}$R, $<:^{-}_{+}$L	<:∃R	$<:^{-}_{+}$L
	∃	<:∃L	<:∃L	<:∃L
	other	$<:^{-}_{+}$R	<:∃R	<:Equiv

  The only case in which there are two possible final rules is in the ∀/∀ case. In this case, regardless of the choice of rule, by inversion we get subderivations $\Gamma \vdash A <:^{-} B \dashv \Delta_1$ and $\Gamma \vdash A <:^{-} B \dashv \Delta_2$.

- If $\Gamma \vdash A <:^{-} B \dashv \Delta_1$ and $\Gamma \vdash A <:^{-} B \dashv \Delta_2$, then we know the last rule ending the derivation of $\mathcal{D}_1$ and $\mathcal{D}_2$ must be:

		B: ∀	B: ∃	B: other
A	∀	<:∀R	<:∀L	<:∀L
	∃	<:∀R	$<:^{+}_{-}$L, $<:^{+}_{-}$R	$<:^{+}_{-}$L
	other	<:∀R	$<:^{+}_{-}$R	<:Equiv

  The only case in which there are two possible final rules is in the ∀/∀ case. In this case, regardless of the choice of rule, by inversion we get subderivations $\Gamma \vdash A <:^{+} B \dashv \Delta_1$ and $\Gamma \vdash A <:^{+} B \dashv \Delta_2$.

As a result, the result follows by a routine induction. □

Theorem 5 (Determinacy of Typing)**.**

(1) Checking: *Given* Γ, e, A, p *such that* $\mathcal{D}_1 :: \Gamma \vdash e \Leftarrow A\ p \dashv \Delta_1$ *and* $\mathcal{D}_2 :: \Gamma \vdash e \Leftarrow A\ p \dashv \Delta_2$, *it is the case that* $\Delta_1 = \Delta_2$.

(2) Synthesis: *Given* Γ, e *such that* $\mathcal{D}_1 :: \Gamma \vdash e \Rightarrow B_1\ p_1 \dashv \Delta_1$ *and* $\mathcal{D}_2 :: \Gamma \vdash e \Rightarrow B_2\ p_2 \dashv \Delta_2$, *it is the case that* $B_1 = B_2$ *and* $p_1 = p_2$ *and* $\Delta_1 = \Delta_2$.

(3) Spine judgments:

Given Γ, e, A, p *such that* $\mathcal{D}_1 :: \Gamma \vdash e : A\ p \gg C_1\ q_1 \dashv \Delta_1$ *and* $\mathcal{D}_2 :: \Gamma \vdash e : A\ p \gg C_2\ q_2 \dashv \Delta_2$, *it is the case that* $C_1 = C_2$ *and* $q_1 = q_2$ *and* $\Delta_1 = \Delta_2$.

The same applies for derivations of the principality-recovering judgments $\Gamma \vdash e : A\ p \gg C_k\ \lceil q_k \rceil \dashv \Delta_k$.

(4) Match judgments:

Given Γ, Π, $\vec{A}$, p, C *such that* $\mathcal{D}_1 :: \Gamma \vdash \Pi :: \vec{A} \Leftarrow C\ p \dashv \Delta_1$ *and* $\mathcal{D}_2 :: \Gamma \vdash \Pi :: \vec{A} \Leftarrow C\ p \dashv \Delta_2$, *it is the case that* $\Delta_1 = \Delta_2$.

Given Γ, P, Π, $\vec{A}$, p, C *such that* $\mathcal{D}_1 :: \Gamma\ /\ P \vdash \Pi :: \vec{A} \Leftarrow C\ p \dashv \Delta_1$ *and* $\mathcal{D}_2 :: \Gamma\ /\ P \vdash \Pi :: \vec{A} \Leftarrow C\ p \dashv \Delta_2$, *it is the case that* $\Delta_1 = \Delta_2$.

Proof.

Proof of Part (1) (checking).

The rules with a checking judgment in the conclusion are: 1I, 1I$\hat{\alpha}$, $\forall$I, $\wedge$I, $\supset$I, $\supset$I$\bot$, $\to$I, $\to$I$\hat{\alpha}$, Rec, $+\mathrm{I}_k$, $+\mathrm{I}\hat{\alpha}_k$, $\times$I, $\times$I$\hat{\alpha}$, Case, Nil, Cons.

The table below shows which rules apply for given e and A. The extra "*chk-I*?" column highlights the role of the "*chk-I*" ("check-intro") category of syntactic forms: we restrict the introduction rules for $\forall$ and $\supset$ to type only these forms. For example, given $e = x$ and $A = (\forall\alpha : \kappa.\, A_0)$, we need not choose between Sub and $\forall$I: the latter is ruled out by its *chk-I* premise.

			A										
		chk-I?	$\forall$	$\supset$ *Note 1*	$\exists$	$\wedge$	$\to$	$+$	$\times$	1	$\hat{\alpha}$	α	Vec
e	$\lambda x.\, e_0$	*chk-I*	$\forall$I	$\supset$I/$\supset$I$\bot$	Sub	$\wedge$I	$\to$I	$\emptyset$	$\emptyset$	$\emptyset$	$\to$I$\hat{\alpha}$	$\emptyset$	$\emptyset$
	rec $x.\, v$	*Note 2*	Rec	Rec	Rec	Rec	Rec	Rec	Rec	Rec	Rec	Rec	$\emptyset$
	$\mathrm{inj}_k\ e_0$	*chk-I*	$\forall$I	$\supset$I/$\supset$I$\bot$	Sub	$\wedge$I	$\emptyset$	$+\mathrm{I}_k$	$\emptyset$	$\emptyset$	$+\mathrm{I}\hat{\alpha}_k$	$\emptyset$	$\emptyset$
	$\langle e_1, e_2 \rangle$	*chk-I*	$\forall$I	$\supset$I/$\supset$I$\bot$	Sub	$\wedge$I	$\emptyset$	$\emptyset$	$\times$I	$\emptyset$	$\times$I$\hat{\alpha}$	$\emptyset$	$\emptyset$
	[]	*chk-I*	$\forall$I	$\supset$I/$\supset$I$\bot$	Sub	$\wedge$I	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	Nil
	$e_1 :: e_2$	*chk-I*	$\forall$I	$\supset$I/$\supset$I$\bot$	Sub	$\wedge$I	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	Cons
	()	*chk-I*	$\forall$I	$\supset$I/$\supset$I$\bot$	Sub	$\wedge$I	$\emptyset$	$\emptyset$	$\emptyset$	1I	1I$\hat{\alpha}$	$\emptyset$	$\emptyset$
	case(e_0, Π)	*Note 3*	Case	Case	Case	Case	Case	Case	Case	Case	Case	Case	Case
	x		Sub	Sub	Sub	Sub	Sub	Sub	Sub	Sub	Sub	Sub	Sub
	$(e_0 : A)$		Sub	Sub	Sub	Sub	Sub	Sub	Sub	Sub	Sub	Sub	Sub
	$e_1\ s$		Sub	Sub	Sub	Sub	Sub	Sub	Sub	Sub	Sub	Sub	Sub

Notes:

- *Note 1:* The choice between $\supset$I and $\supset$I$\bot$ is resolved by Lemma 81 (Determinacy of Auxiliary Judgments) (5).
- *Note 2:* Fixed points are a checking form, but not an introduction form. So if e is rec $x.\, v$, we need not choose between an introduction rule for a large connective and the Rec rule: only the Rec rule is viable. Large connectives must, therefore, be introduced *inside* the typing of the body v.
- *Note 3:* Case expressions are a checking form, but not an introduction form. So if e is a case expression, we need not choose between an introduction rule for a large connective and the Case rule: only the Case rule is viable. Large connectives must, therefore, be introduced *inside* the branches.

Proof of Part (2) (synthesis). Only four rules have a synthesis judgment in the conclusion: Var, Anno, $\to$E, and $\to$E-**!**. Rule Var applies if and only if e has the form x. Rule Anno applies if and only if e has the form $(e_0 : A)$.

Otherwise, the judgment can be derived only if e has the form $e_1\ e_2$, by $\to$E or $\to$E-**!**. If $\mathcal{D}_1$ and $\mathcal{D}_2$ both end in $\to$E or $\to$E-**!**, we are done. Suppose $\mathcal{D}_1$ ends in $\to$E and $\mathcal{D}_2$ ends in $\to$E-**!**. By i.h., the p in the first subderivation of $\to$E must be equal to the one in the first subderivation of $\to$E-**!**, that is, $p = !$. Thus the inputs to the respective second subderivations match, so by i.h. their outputs match; in particular, $q = \not!$. However, from the condition in $\to$E, it must be the case that $\mathsf{FEV}([\Delta]C) \neq \emptyset$, which contradicts the condition $\mathsf{FEV}([\Delta]C) = \emptyset$ in $\to$E-**!**.

Proof of Part (3) (spine judgments). For the ordinary spine judgment, rule EmptySpine applies if and only if the given spine is empty. Otherwise, the choice of rule is determined by the head constructor of the input type: $\to/\to$Spine; $\forall/\forall$Spine; $\supset/\supset$Spine; $\hat{\alpha}/\hat{\alpha}$Spine.

For the principality-recovering spine judgment: If $p = \not!$, only rule SpinePass applies. If $p = !$ and $q = !$, only rule SpinePass applies. If $p = !$ and $q = \not!$, then the rule is determined by $\mathsf{FEV}(C)$: if $\mathsf{FEV}(C) = \emptyset$ then only SpineRecover applies; otherwise, $\mathsf{FEV}(C) \neq \emptyset$ and only SpinePass applies.

Proof of Part (4) (matching). First, the elimination judgment form $\Gamma \ / \ P \vdash \ldots$: It cannot be the case that both $\Gamma \ / \ \sigma \doteq t : \kappa \dashv \bot$ and $\Gamma \ / \ \sigma \doteq t : \kappa \dashv \Theta$, so either Match$\bot$ concludes both $\mathcal{D}_1$ and $\mathcal{D}_2$ (and the result follows), or MatchUnify concludes both $\mathcal{D}_1$ and $\mathcal{D}_2$ (in which case, apply the i.h.).

Now the main judgment form, without "/ P": either Π is empty, or has length one, or has length greater than one. MatchEmpty applies if and only if Π is empty, and MatchSeq applies if and only if Π has length greater than one. So in the rest of this part, we assume Π has length one.

Moreover, MatchBase applies if and only if $\vec{A}$ has length zero. So in the rest of this part, we assume the length of $\vec{A}$ is at least one.

Let A be the first type in $\vec{A}$. Inspection of the rules shows that given particular A and ρ, where ρ is the first pattern, only a single rule can apply, or no rule ("$\emptyset$") can apply, as shown in the following table:

		A					
		$\exists$	$\wedge$	$+$	$\times$	Vec	other
ρ	$\mathsf{inj}_k\ \rho_0$	Match$\exists$	Match$\wedge$	Match$+_k$	$\emptyset$	$\emptyset$	$\emptyset$
	$\langle \rho_1, \rho_2 \rangle$	Match$\exists$	Match$\wedge$	$\emptyset$	Match$\times$	$\emptyset$	$\emptyset$
	z	Match$\exists$	Match$\wedge$	MatchNeg	MatchNeg	MatchNeg	MatchNeg
	$\_$	Match$\exists$	Match$\wedge$	MatchWild	MatchWild	MatchWild	MatchWild
	$[]$	Match$\exists$	Match$\wedge$	$\emptyset$	$\emptyset$	MatchNil	$\emptyset$
	$\rho_1 :: \rho_2$	Match$\exists$	Match$\wedge$	$\emptyset$	$\emptyset$	MatchCons	$\emptyset$

□

K′ Soundness

K′.1 Instantiation

Lemma 83 (Soundness of Instantiation)**.**
If $\Gamma \vdash \hat{\alpha} := \tau : \kappa \dashv \Delta$ *and* $\hat{\alpha} \notin \mathrm{FV}([\Gamma]\tau)$ *and* $[\Gamma]\tau = \tau$ *and* $\Delta \longrightarrow \Omega$ *then* $[\Omega]\hat{\alpha} = [\Omega]\tau$.

Proof. By induction on the derivation of $\Gamma \vdash \hat{\alpha} := \tau : \kappa \dashv \Delta$.

- **Case**

$$\frac{\Gamma_0 \vdash \tau : \kappa}{\Gamma_0, \hat{\alpha} : \kappa, \Gamma_1 \vdash \hat{\alpha} := \tau : \kappa \dashv \underbrace{\Gamma_0, \hat{\alpha} : \kappa = \tau, \Gamma_1}_{\Delta}}\ \mathsf{InstSolve}$$

  $[\Delta]\hat{\alpha} = [\Delta]\tau$ By definition

  ☞ $[\Omega]\hat{\alpha} = [\Omega]\tau$ By Lemma 29 (Substitution Monotonicity) to each side

- **Case**

$$\frac{\hat{\beta} \in \mathsf{unsolved}(\Gamma[\hat{\alpha} : \kappa][\hat{\beta} : \kappa])}{\Gamma[\hat{\alpha} : \kappa][\hat{\beta} : \kappa] \vdash \hat{\alpha} := \underbrace{\hat{\beta}}_{\tau} : \kappa \dashv \underbrace{\Gamma[\hat{\alpha} : \kappa][\hat{\beta} : \kappa = \hat{\alpha}]}_{\Delta}}\ \mathsf{InstReach}$$

$$
\begin{array}{rll}
[\Delta]\hat{\beta} = [\Delta]\hat{\alpha} & & \text{By definition} \\
[\Omega][\Delta]\hat{\beta} = [\Omega][\Delta]\hat{\alpha} & & \text{Applying } \Omega \text{ to each side} \\
\text{☞}\quad [\Omega]\underbrace{\hat{\beta}}_{\tau} = [\Omega]\hat{\alpha} & & \text{By Lemma 29 (Substitution Monotonicity) to each side}
\end{array}
$$

- **Case**

$$\frac{\overbrace{\Gamma_0[\hat{\alpha}_2 : \star, \hat{\alpha}_1 : \star, \hat{\alpha} : \star = \hat{\alpha}_1 \oplus \hat{\alpha}_2]}^{\Gamma'} \vdash\ \hat{\alpha}_1 := \tau_1 : \star \dashv \Theta \qquad \Theta \vdash\ \hat{\alpha}_2 := [\Theta]\tau_2 : \star \dashv \Delta}{\Gamma_0[\hat{\alpha} : \star] \vdash\ \hat{\alpha} := \tau_1 \oplus \tau_2 : \star \dashv \Delta}\ \textsf{InstBin}$$

$$
\begin{array}{rll}
\Delta \longrightarrow \Omega & & \text{Given} \\
\Gamma' \vdash \hat{\alpha}_1 := \tau_1 : \star \dashv \Theta & & \text{Subderivation} \\
\Theta \longrightarrow \Delta & & \text{By Lemma 43 (Instantiation Extension)} \\
\Theta \longrightarrow \Omega & & \text{By Lemma 33 (Extension Transitivity)} \\
[\Omega]\hat{\alpha}_1 = [\Omega]\tau_1 & & \text{By i.h.} \\
\\
\Theta \vdash \hat{\alpha}_2 := [\Theta]\tau_2 : \star \dashv \Delta & & \text{Subderivation} \\
[\Omega]\hat{\alpha}_2 = [\Omega][\Theta]\tau_2 & & \text{By i.h.} \\
= [\Omega]\tau_2 & & \text{By Lemma 29 (Substitution Monotonicity)} \\
\\
([\Omega]\tau_1) \oplus ([\Omega]\tau_2) = ([\Omega]\hat{\alpha}_1) \oplus ([\Omega]\hat{\alpha}_2) & & \text{By above equalities} \\
= [\Omega](\hat{\alpha}_1 \oplus \hat{\alpha}_2) & & \text{By definition of substitution} \\
= [\Omega]([\Gamma']\hat{\alpha}) & & \text{By definition of substitution} \\
= [\Omega]\hat{\alpha} & & \text{By Lemma 29 (Substitution Monotonicity)} \\
\\
\text{☞}\quad [\Omega]\underbrace{(\tau_1 \oplus \tau_2)}_{\tau} = [\Omega]\hat{\alpha} & & \text{By definition of substitution}
\end{array}
$$

- **Case**

$$\frac{}{\Gamma_0[\hat{\alpha} : \mathbb{N}] \vdash\ \hat{\alpha} := \mathsf{zero} : \mathbb{N} \dashv \Gamma_0[\hat{\alpha} : \mathbb{N} = \mathsf{zero}]}\ \textsf{InstZero}$$

  Similar to the InstSolve case.

- **Case** $\dfrac{\Gamma_0[\hat{\alpha}_1 : \mathbb{N}, \hat{\alpha} : \mathbb{N} = \mathsf{succ}(\hat{\alpha}_1)] \vdash\ \hat{\alpha}_1 := t_1 : \mathbb{N} \dashv \Delta}{\Gamma_0[\hat{\alpha} : \mathbb{N}] \vdash\ \hat{\alpha} := \mathsf{succ}(t_1) : \mathbb{N} \dashv \Delta}\ \textsf{InstSucc}$

  Similar to the InstBin case, but simpler. □

Lemma 84 (Soundness of Checkeq)**.**
If $\Gamma \vdash\ \sigma \mathrel{\mathring{=}} t : \kappa \dashv \Delta$ *where* $\Delta \longrightarrow \Omega$ *then* $[\Omega]\sigma = [\Omega]t$.

Proof. By induction on the given derivation.

- **Case**

$$\frac{}{\Gamma \vdash\ u \mathrel{\mathring{=}} u : \kappa \dashv \Gamma}\ \textsf{CheckeqVar}$$

  ☞ $[\Omega]u = [\Omega]u$ By reflexivity of equality

- **Cases** CheckeqUnit, CheckeqZero: Similar to the CheckeqVar case.

- **Case**

$$\frac{\Gamma \vdash\ \sigma_0 \mathrel{\mathring{=}} t_0 : \mathbb{N} \dashv \Delta}{\Gamma \vdash\ \mathsf{succ}(\sigma_0) \mathrel{\mathring{=}} \mathsf{succ}(t_0) : \mathbb{N} \dashv \Delta}\ \textsf{CheckeqSucc}$$

$$
\begin{array}{rll}
\Gamma \vdash \sigma_0 \mathrel{\mathring{=}} t_0 : \mathbb{N} \dashv \Delta & & \text{Subderivation} \\
[\Omega]\sigma_0 = [\Omega]t_0 & & \text{By i.h.} \\
\mathsf{succ}([\Omega]\sigma_0) = \mathsf{succ}([\Omega]t_0) & & \text{By congruence} \\
\text{☞}\quad [\Omega](\mathsf{succ}(\sigma_0)) = [\Omega](\mathsf{succ}(t_0)) & & \text{By definition of substitution}
\end{array}
$$

- **Case** $\dfrac{\Gamma \vdash\ \sigma_0 \doteq t_0 : \star \dashv \Theta \qquad \Theta \vdash\ [\Theta]\sigma_1 \doteq [\Theta]t_1 : \star \dashv \Delta}{\Gamma \vdash\ \sigma_0 \oplus \sigma_1 \doteq t_0 \oplus t_1 : \star \dashv \Delta}\ \mathsf{CheckeqBin}$

$\Gamma \vdash\ \sigma_0 \doteq t_0 : \mathbb{N} \dashv \Delta$	Subderivation
$\Theta \vdash\ [\Theta]\sigma_1 \doteq [\Theta]t_1 : \star \dashv \Delta$	Subderivation
$\Delta \longrightarrow \Omega$	Given
$\Theta \longrightarrow \Delta$	By Lemma 46 (Checkeq Extension)
$\Theta \longrightarrow \Omega$	By Lemma 33 (Extension Transitivity)
$[\Omega]\sigma_0 = [\Omega]t_0$	By i.h. on first subderivation
$[\Omega][\Theta]\sigma_1 = [\Omega][\Theta]t_1$	By i.h. on second subderivation
$[\Omega][\Theta]\sigma_1 = [\Omega]\sigma_1$	By Lemma 29 (Substitution Monotonicity)
$[\Omega][\Theta]t_1 = [\Omega]t_1$	By Lemma 29 (Substitution Monotonicity)
$[\Omega]\sigma_1 = [\Omega]t_1$	By transitivity of equality
$[\Omega]\sigma_0 \oplus [\Omega]\sigma_1 = [\Omega]t_0 \oplus [\Omega]t_1$	By congruence of equality
☞ $[\Omega](\sigma_0 \oplus \sigma_1) = [\Omega](t_0 \oplus t_1)$	By definition of substitution

- **Case** $\dfrac{\Gamma[\hat{\alpha}] \vdash\ \hat{\alpha} := t : \kappa \dashv \Delta \qquad \hat{\alpha} \notin \mathrm{FV}(t)}{\Gamma[\hat{\alpha}] \vdash\ \hat{\alpha} \doteq t : \kappa \dashv \Delta}\ \mathsf{CheckeqInstL}$

$\Gamma[\hat{\alpha}] \vdash\ \hat{\alpha} := t : \kappa \dashv \Delta$	Subderivation
$\hat{\alpha} \notin \mathrm{FV}(t)$	Premise
☞ $[\Omega]\hat{\alpha} = [\Omega]t$	By Lemma 83 (Soundness of Instantiation)

- **Case** $\dfrac{\Gamma[\hat{\alpha} : \kappa] \vdash\ \hat{\alpha} := \sigma : \kappa \dashv \Delta \qquad \hat{\alpha} \notin \mathrm{FV}(t)}{\Gamma[\hat{\alpha} : \kappa] \vdash\ \sigma \doteq \hat{\alpha} : \kappa \dashv \Delta}\ \mathsf{CheckeqInstR}$

  Similar to the CheckeqInstL case. □

Lemma 85 (Soundness of Propositional Equivalence)**.**
If $\Gamma \vdash\ P \equiv Q \dashv \Delta$ *where* $\Delta \longrightarrow \Omega$ *then* $[\Omega]P = [\Omega]Q$.

Proof. By induction on the given derivation.

- **Case** $\dfrac{\Gamma \vdash\ \sigma_1 \doteq t_1 : \mathbb{N} \dashv \Theta \qquad \Theta \vdash\ [\Theta]\sigma_2 \doteq [\Theta]t_2 : \mathbb{N} \dashv \Delta}{\Gamma \vdash\ (\sigma_1 = \sigma_2) \equiv (t_1 = t_2) \dashv \Delta}\ \equiv\mathsf{PropEq}$

$\Delta \longrightarrow \Omega$	Given
$\Theta \longrightarrow \Delta$	By Lemma 46 (Checkeq Extension) (on 2nd premise)
$\Theta \longrightarrow \Omega$	By Lemma 33 (Extension Transitivity)
$\Gamma \vdash\ \sigma_1 \doteq t_1 : \mathbb{N} \dashv \Theta$	Given
$[\Omega]\sigma_1 = [\Omega]t_1$	By Lemma 84 (Soundness of Checkeq)
$\Theta \vdash\ [\Theta]\sigma_2 \doteq [\Theta]t_2 : \mathbb{N} \dashv \Delta$	Given
$[\Omega][\Theta]\sigma_2 = [\Omega][\Theta]t_2$	By Lemma 84 (Soundness of Checkeq)
$[\Omega][\Theta]\sigma_2 = [\Omega]\sigma_2$	By Lemma 29 (Substitution Monotonicity)
$[\Omega][\Theta]t_2 = [\Omega]t_2$	By Lemma 29 (Substitution Monotonicity)
$[\Omega]\sigma_2 = [\Omega]t_2$	By transitivity of equality
$([\Omega]\sigma_1 = [\Omega]\sigma_2) = ([\Omega]t_1 = [\Omega]t_2)$	By congruence of equality
☞ $[\Omega](\sigma_1 = \sigma_2) = [\Omega](t_1 = t_2)$	By definition of substitution

□

Lemma 86 (Soundness of Algorithmic Equivalence)**.**
If $\Gamma \vdash\ A \equiv B \dashv \Delta$ *where* $\Delta \longrightarrow \Omega$ *then* $[\Omega]A = [\Omega]B$.

Proof. By induction on the given derivation.

- **Case**

$$\frac{}{\Gamma \vdash\ \alpha \equiv \alpha \dashv \Gamma}\ \equiv\mathsf{Var}$$

  ☞ $[\Omega]\alpha = [\Omega]\alpha$ By reflexivity of equality

- **Cases** ≡Exvar, ≡Unit: Similar to the ≡Var case.

- **Case** $\dfrac{\Gamma \vdash A_1 \equiv B_1 \dashv \Theta \qquad \Theta \vdash [\Theta]A_2 \equiv [\Theta]B_2 \dashv \Delta}{\Gamma \vdash A_1 \oplus A_2 \equiv B_1 \oplus B_2 \dashv \Delta} \equiv\oplus$

$\Delta \longrightarrow \Omega$	Given
$\Theta \vdash [\Theta]A_2 \equiv [\Theta]B_2 \dashv \Delta$	Subderivation
$\Theta \longrightarrow \Delta$	By Lemma 49 (Equivalence Extension)
$\Theta \longrightarrow \Omega$	By Lemma 33 (Extension Transitivity)
$\Gamma \vdash A_1 \equiv B_1 \dashv \Theta$	Subderivation
$[\Omega]A_1 = [\Omega]B_1$	By i.h.
$\Delta \longrightarrow \Omega$	Given
$[\Omega][\Theta]A_2 = [\Omega][\Theta]B_2$	By i.h.
$[\Omega]A_2 = [\Omega]B_2$	By Lemma 29 (Substitution Monotonicity)
☞ $([\Omega]A_1) \oplus ([\Omega]A_2) = ([\Omega]B_1) \oplus ([\Omega]B_2)$	By above equations

- **Case** $\dfrac{\Gamma, \alpha : \kappa \vdash A_0 \equiv B_0 \dashv \Delta, \alpha : \kappa, \Delta'}{\Gamma \vdash \forall \alpha : \kappa.\, A_0 \equiv \forall \alpha : \kappa.\, B_0 \dashv \Delta} \equiv\forall$

$\Gamma, \alpha : \kappa \vdash A_0 \equiv B_0 \dashv \Delta, \alpha : \kappa, \Delta'$	Subderivation
$\Delta \longrightarrow \Omega$	Given
$\Gamma, \alpha : \kappa, \cdot \longrightarrow \Delta, \alpha : \kappa, \Delta'$	By Lemma 49 (Equivalence Extension)
Δ' soft	Since $\cdot$ is soft
$\Delta, \alpha : \kappa, \Delta' \longrightarrow \Omega, \alpha : \kappa, \Omega_Z$	By Lemma 24 (Soft Extension)
$\Gamma, \alpha : \kappa \vdash A_0$ *type*	By validity on subderivation
$\Gamma, \alpha : \kappa \vdash B_0$ *type*	By validity on subderivation
$FV(A_0) \subseteq \mathsf{dom}(\Gamma, \alpha : \kappa)$	By well-typing of A_0
$FV(B_0) \subseteq \mathsf{dom}(\Gamma, \alpha : \kappa)$	By well-typing of B_0
$\Gamma, \alpha : \kappa \longrightarrow \Omega, \alpha : \kappa$	By $\longrightarrow$Uvar
$FV(A_0) \subseteq \mathsf{dom}(\Omega, \alpha : \kappa)$	By Lemma 20 (Declaration Order Preservation)
$FV(B_0) \subseteq \mathsf{dom}(\Omega, \alpha : \kappa)$	By Lemma 20 (Declaration Order Preservation)
$[\Omega, \alpha : \kappa, \Omega_Z]A_0 = [\Omega, \alpha : \kappa]A_0$	By definition of substitution, since $FV(A_0) \cap \mathsf{dom}(\Omega_Z) = \emptyset$
$[\Omega, \alpha : \kappa, \Omega_Z]B_0 = [\Omega, \alpha : \kappa]B_0$	By definition of substitution, since $FV(B_0) \cap \mathsf{dom}(\Omega_Z) = \emptyset$
$[\Omega, \alpha : \kappa]A_0 = [\Omega, \alpha : \kappa]B_0$	By transitivity of equality
$[\Omega]A_0 = [\Omega]B_0$	From definition of substitution
$\forall \alpha : \kappa.\, [\Omega]A_0 = \forall \alpha : \kappa.\, [\Omega]B_0$	Adding quantifier to each side
$[\Omega](\forall \alpha : \kappa.\, A_0) = [\Omega](\forall \alpha : \kappa.\, B_0)$	By definition of subsitution

- **Case** $\dfrac{\Gamma \vdash P \equiv Q \dashv \Theta \qquad \Theta \vdash [\Theta]A_0 \equiv [\Theta]B_0 \dashv \Delta}{\Gamma \vdash P \supset A_0 \equiv Q \supset B_0 \dashv \Delta} \equiv\supset$

$\Delta \longrightarrow \Omega$	Given
$\Theta \vdash [\Theta]A_0 \equiv [\Theta]B_0 \dashv \Delta$	Subderivation
$\Theta \longrightarrow \Delta$	By Lemma 49 (Equivalence Extension)
$\Theta \longrightarrow \Omega$	By Lemma 33 (Extension Transitivity)
$\Gamma \vdash P \equiv Q \dashv \Theta$	Subderivation
$[\Omega]P = [\Omega]Q$	By Lemma 85 (Soundness of Propositional Equivalence)
$\Theta \vdash [\Theta]A_0 \equiv [\Theta]B_0 \dashv \Delta$	Subderivation
$[\Omega][\Theta]A_0 = [\Omega][\Theta]B_0$	By i.h.
$[\Omega]A_0 = [\Omega]B_0$	By Lemma 29 (Substitution Monotonicity)

- **Case** $\dfrac{\Gamma \vdash\ P \equiv Q\ \dashv \Theta \qquad \Theta \vdash\ [\Theta]A_0 \equiv [\Theta]B_0\ \dashv \Delta}{\Gamma \vdash\ A_0 \wedge P \equiv B_0 \wedge Q\ \dashv \Delta}\ \equiv\!\wedge$

  Similar to the $\equiv\!\supset$ case.

- **Case** $\dfrac{\Gamma[\hat\alpha] \vdash\ \hat\alpha := \tau : \star\ \dashv \Delta \qquad \hat\alpha \notin FV(\tau)}{\Gamma[\hat\alpha] \vdash\ \hat\alpha \equiv \underbrace{\tau}_{A}\ \dashv \Delta}\ \equiv\mathsf{InstantiateL}$

	$\Gamma[\hat\alpha] \vdash\ \hat\alpha := \tau : \star\ \dashv \Delta$	Subderivation
☞	$[\Omega]\hat\alpha = [\Omega]\tau$	By Lemma 83 (Soundness of Instantiation)

- **Case** $\equiv$InstantiateR: Similar to the $\equiv$InstantiateL case. □

K′.2 Soundness of Checkprop

Lemma 87 (Soundness of Checkprop)**.**
If $\Gamma \vdash\ P$ *true* $\dashv \Delta$ *and* $\Delta \longrightarrow \Omega$ *then* $\Psi \vdash\ [\Omega]P$ *true.*

Proof. By induction on the derivation of $\Gamma \vdash\ P$ *true* $\dashv \Delta$.

- **Case** $\dfrac{\Gamma \vdash\ \sigma \doteq t : \mathbb{N}\ \dashv \Delta}{\Gamma \vdash\ \underbrace{\sigma = t}_{P}\ \textit{true}\ \dashv \Delta}\ \mathsf{CheckpropEq}$

	$\Gamma \vdash\ \sigma \doteq t : \mathbb{N}\ \dashv \Delta$	Subderivation
	$[\Omega]\sigma = [\Omega]t$	By Lemma 84 (Soundness of Checkeq)
	$\Psi \vdash\ [\Omega]\sigma = [\Omega]t$ *true*	By DeclCheckpropEq
	$\Psi \vdash\ [\Omega](\sigma = t)$ *true*	By def. of subst.
☞	$\Psi \vdash\ [\Omega]P$ *true*	By $P = (\sigma = t)$

  □

K′.3 Soundness of Eliminations (Equality and Proposition)

Lemma 88 (Soundness of Equality Elimination)**.**
If $[\Gamma]\sigma = \sigma$ *and* $[\Gamma]t = t$ *and* $\Gamma \vdash\ \sigma : \kappa$ *and* $\Gamma \vdash\ t : \kappa$ *and* $FEV(\sigma) \cup FEV(t) = \emptyset$*, then:*

(1) If $\Gamma\ /\ \sigma \doteq t : \kappa\ \dashv \Delta$
then $\Delta = (\Gamma, \Theta)$ *where* $\Theta = (\alpha_1 = t_1, \ldots, \alpha_n = t_n)$ *and*
for all Ω *such that* $\Gamma \longrightarrow \Omega$
and all t' *such that* $\Omega \vdash\ t' : \kappa'$*,*
it is the case that $[\Omega, \Theta]t' = [\theta][\Omega]t'$*, where* $\theta = mgu(\sigma, t)$*.*

(2) If $\Gamma\ /\ \sigma \doteq t : \kappa\ \dashv \bot$ *then* $mgu(\sigma, t) = \bot$ *(that is, no most general unifier exists).*

Proof. First, we need to recall a few properties of term unification.

(i) If σ is a term, then $mgu(\sigma, \sigma) = id$.

(ii) If f is a unary constructor, then $mgu(f(\sigma), f(t)) = mgu(\sigma, t)$, supposing that $mgu(\sigma, t)$ exists.

(iii) If f is a binary constructor, and $\sigma = mgu(f(\sigma_1, \sigma_2), f(t_1, t_2))$ and $\sigma_1 = mgu(\sigma_1, t_1)$ and $\sigma_2 = mgu([\sigma_1]\sigma_2, [\sigma_1]t_2)$, then $\sigma = \sigma_2 \circ \sigma_1 = \sigma_1 \circ \sigma_2$.

(iv) If $\alpha \notin FV(t)$, then $mgu(\alpha, t) = (\alpha = t)$.

(v) If f is an n-ary constructor, and σ_i and t_i (for $i \leq n$) have no unifier, then $f(\sigma_1, \ldots, \sigma_n)$ and $f(t_1, \ldots, t_n)$ have no unifier.

We proceed by induction on the derivation of $\Gamma\ /\ \sigma \doteq t : \kappa\ \dashv \Delta^{\bot}$, proving both parts with a single induction.

- **Case**
$$\frac{}{\Gamma \mathrel{/} \alpha \doteq \alpha : \kappa \dashv \Gamma}\ \mathsf{ElimeqUvarRefl}$$
Here we have $\Delta = \Gamma$, so we are in part (1).
Let $\theta = id$ (which is $\mathsf{mgu}(\sigma, \sigma)$).
We can easily show $[id][\Omega]\alpha = [\Omega, \alpha] = [\Omega, \cdot]\alpha$.

- **Case**
$$\frac{}{\Gamma \mathrel{/} \mathsf{zero} \doteq \mathsf{zero} : \mathbb{N} \dashv \Gamma}\ \mathsf{ElimeqZero}$$
Similar to the ElimeqUvarRefl case.

- **Case**
$$\frac{\Gamma \mathrel{/} t_1 \doteq t_2 : \mathbb{N} \dashv \Delta^\perp}{\Gamma \mathrel{/} \mathsf{succ}(t_1) \doteq \mathsf{succ}(t_2) : \mathbb{N} \dashv \Delta^\perp}\ \mathsf{ElimeqSucc}$$
We distinguish two subcases:
  - **Case** $\Delta^\perp = \Delta$:
    Since we have the same output context in the conclusion and premise, the "for all $t' \ldots$" part follows immediately from the i.h. (1).
    The i.h. also gives us $\theta_0 = \mathsf{mgu}(t_1, t_2)$.
    Let $\theta = \theta_0$. By property (ii), $\mathsf{mgu}(t_1, t_2) = \mathsf{mgu}(\mathsf{succ}(t_1), \mathsf{succ}(t_2)) = \theta$.
  - **Case** $\Delta^\perp = \perp$:
$$\begin{array}{rll} & \Gamma \mathrel{/} t_1 \doteq t_2 : \mathbb{N} \dashv \perp & \text{Subderivation} \\ \mathsf{mgu}(t_1, t_2) = \perp & & \text{By i.h. (2)} \\ \text{☞}\quad \mathsf{mgu}(\mathsf{succ}(t_1), \mathsf{succ}(t_2)) = \perp & & \text{By contrapositive of property (ii)} \end{array}$$

- **Case**
$$\frac{\alpha \notin \mathrm{FV}(t) \qquad (\alpha = -) \notin \Gamma}{\Gamma \mathrel{/} \alpha \doteq t : \kappa \dashv \Gamma, \alpha = t}\ \mathsf{ElimeqUvarL}$$
Here $\Delta \neq \perp$, so we are in part (1).
$$\begin{array}{rll} [\Omega, \alpha = t]t' = & \big[[\Omega]t/\alpha\big][\Omega]t' & \text{By a property of substitution} \\ = & [\Omega][t/\alpha][\Omega]t' & \text{By a property of substitution} \\ = & [\Omega][\theta][\Omega]t' & \text{By } \mathsf{mgu}(\alpha, t) = (\alpha/t) \\ \text{☞}\quad = & [\theta][\Omega]t' & \text{By a property of substitution } (\theta \text{ creates no evars}) \end{array}$$

- **Case**
$$\frac{\alpha \notin \mathrm{FV}(t) \qquad (\alpha = -) \notin \Gamma}{\Gamma \mathrel{/} t \doteq \alpha : \kappa \dashv \Gamma, \alpha = t}\ \mathsf{ElimeqUvarR}$$
Similar to the ElimeqUvarL case.

- **Case**
$$\frac{}{\Gamma \mathrel{/} 1 \doteq 1 : \star \dashv \Gamma}\ \mathsf{ElimeqUnit}$$
Similar to the ElimeqUvarRefl case.

- **Case**
$$\frac{\Gamma \mathrel{/} \tau_1 \doteq \tau_1' : \star \dashv \Theta \qquad \Theta \mathrel{/} [\Theta]\tau_1 \doteq [\Theta]\tau_2' : \star \dashv \Delta^\perp}{\Gamma \mathrel{/} \tau_1 \oplus \tau_2 \doteq \tau_1' \oplus \tau_2' : \star \dashv \Delta^\perp}\ \mathsf{ElimeqBin}$$
Either $\Delta^\perp$ is some Δ, or it is $\perp$.
  - **Case** $\Delta^\perp = \Delta$:

$$\begin{array}{rll} & \Gamma \;/\; \tau_1 \stackrel{\circ}{=} \tau_1' : \star \dashv \Theta & \text{Subderivation} \\ & \Theta = (\Gamma, \Delta_1) & \text{By i.h. (1)} \\ \text{(IH-1st)} & [\Omega, \Delta_1]u_1 = [\theta_1][\Omega]u_1 & '' \text{ for all } \Omega \vdash u_1 : \kappa' \\ & \theta_1 = \mathsf{mgu}(\tau_1, \tau_1') & '' \end{array}$$

$$\begin{array}{rll} & \Theta \;/\; [\Theta]\tau_1 \stackrel{\circ}{=} [\Theta]\tau_2' : \star \dashv \Delta & \text{Subderivation} \\ & \Delta = (\Theta, \Delta_2) & \text{By i.h. (1)} \\ \text{(IH-2nd)} & [\Omega, \Delta_1, \Delta_2]u_2 = [\theta_2][\Omega, \Delta_1]u_2 & '' \text{ for all } \Omega \vdash u_2 : \kappa' \\ & \theta_2 = \mathsf{mgu}(\tau_2, \tau_2') & '' \end{array}$$

Suppose $\Omega \vdash u : \kappa'$.

$$\begin{array}{rll} [\Omega, \Delta_1, \Delta_2]u = [\theta_2][\Omega, \Delta_1]u & & \text{By (IH-2nd), with } u_2 = u \\ = [\theta_2][\theta_1][\Omega]u & & \text{By (IH-1st), with } u_1 = u \\ \text{☞} \quad = [\Omega][\theta_2 \circ \theta_1]u & & \text{By a property of substitution} \end{array}$$

$$\text{☞} \quad \theta_2 \circ \theta_1 = \mathsf{mgu}((\tau_1 \oplus \tau_2), (\tau_1' \oplus \tau_2')) \quad \text{By property (iii) of substitution}$$

– **Case** $\Delta^{\perp} = \perp$:
Use the i.h. (2) on the second premise to show $\mathsf{mgu}(\tau_2, \tau_2') = \perp$, then use property (v) of unification to show $\mathsf{mgu}((\tau_1 \oplus \tau_2), (\tau_1' \oplus \tau_2')) = \perp$.

- **Case** $$\frac{\Gamma \;/\; \tau_1 \stackrel{\circ}{=} \tau_1' : \star \dashv \perp}{\Gamma \;/\; \tau_1 \oplus \tau_2 \stackrel{\circ}{=} \tau_1' \oplus \tau_2' : \star \dashv \perp}\ \mathsf{ElimeqBinBot}$$

  Similar to the $\perp$ subcase for ElimeqSucc, but using property (v) instead of property (ii).

- **Case** $$\frac{\sigma \# t}{\Gamma \;/\; \sigma \stackrel{\circ}{=} t : \kappa \dashv \perp}\ \mathsf{ElimeqClash}$$

  Since $\sigma \# t$, we know σ and t have different head constructors, and thus no unifier. □

Theorem 6 (Soundness of Algorithmic Subtyping)**.**
If $[\Gamma]A = A$ *and* $[\Gamma]B = B$ *and* $\Gamma \vdash A$ *type and* $\Gamma \vdash B$ *type and* $\Delta \longrightarrow \Omega$ *and* $\Gamma \vdash A <:^{\pm} B \dashv \Delta$ *then* $[\Omega]\Delta \vdash [\Omega]A \leq^{\pm} [\Omega]B$.

Proof. By induction on the given derivation.

- **Case** $\dfrac{B \text{ not headed by } \forall \qquad \Gamma, \blacktriangleright_{\hat\alpha}, \hat\alpha : \kappa \vdash [\hat\alpha/\alpha]A_0 <:^{-} B \dashv \Delta, \blacktriangleright_{\hat\alpha}, \Theta}{\Gamma \vdash \forall \alpha : \kappa.\, A_0 <:^{-} B \dashv \Delta}\, <:\forall L$

  Let $\Omega' = (\Omega, |\blacktriangleright_{\hat\alpha}, \Theta|)$.

$\Gamma, \blacktriangleright_{\hat\alpha}, \hat\alpha : \kappa \vdash [\hat\alpha/\alpha]A_0 <:^{-} B \dashv \Delta, \blacktriangleright_{\hat\alpha}, \Theta$	Subderivation
$\Delta \longrightarrow \Omega$	Given
$(\Delta, \blacktriangleright_{\hat\alpha}, \Theta) \longrightarrow \Omega'$	By Lemma 25 (Filling Completes)
$\Gamma \vdash \forall \alpha : \kappa.\, A_0$ *type*	Given
$\Gamma, \alpha : \kappa \vdash A_0$ *type*	By inversion (ForallWF)
$\Gamma, \blacktriangleright_{\hat\alpha}, \hat\alpha : \kappa \vdash [\hat\alpha/\alpha]A_0$ *type*	By a property of substitution
$\Gamma \vdash B$ *type*	Given
$[\Omega'](\Delta, \blacktriangleright_{\hat\alpha}, \Theta) \vdash [\Omega'][\hat\alpha/\alpha]A_0 \leq^{-} [\Omega']B$	By i.h.
$\Omega \vdash B$ *type*	By Lemma 36 (Extension Weakening (Sorts))
$[\Omega']B = [\Omega]B$	By Lemma 17 (Substitution Stability)
$[\Omega'](\Delta, \blacktriangleright_{\hat\alpha}, \Theta) \vdash [\Omega'][\hat\alpha/\alpha]A_0 \leq^{-} [\Omega]B$	By above equality
$[\Omega'](\Delta, \blacktriangleright_{\hat\alpha}, \Theta) \vdash \big[[\Omega']\hat\alpha/\alpha\big][\Omega']A_0 \leq^{-} [\Omega]B$	By distributivity of substitution
$\Gamma, \blacktriangleright_{\hat\alpha}, \hat\alpha : \kappa \vdash \hat\alpha : \kappa$	By VarSort
$\Gamma, \blacktriangleright_{\hat\alpha}, \hat\alpha : \kappa \longrightarrow \Delta, \blacktriangleright_{\hat\alpha}, \Theta$	By Lemma 50 (Subtyping Extension)
Θ is soft	By Lemma 22 (Extension Inversion) (ii)
$\Delta, \blacktriangleright_{\hat\alpha}, \Theta \vdash \hat\alpha : \kappa$	By Lemma 36 (Extension Weakening (Sorts))
$(\Delta, \blacktriangleright_{\hat\alpha}, \Theta) \longrightarrow \Omega'$	Above
$[\Omega']\Omega' \vdash [\Omega']\hat\alpha : \kappa$	By Lemma 14 (Substitution for Sorting)
$[\Omega'](\Delta, \blacktriangleright_{\hat\alpha}, \Theta) \vdash [\Omega']\hat\alpha : \kappa$	By Lemma 54 (Completing Stability)
$[\Omega'](\Delta, \blacktriangleright_{\hat\alpha}, \Theta) \vdash \forall \alpha : \kappa.\, [\Omega']A_0 \leq^{-} [\Omega]B$	By $\leq\forall L$
$[\Omega'](\Delta, \blacktriangleright_{\hat\alpha}, \Theta) \vdash \forall \alpha : \kappa.\, [\Omega, \alpha : \kappa]A_0 \leq^{-} [\Omega]B$	By Lemma 17 (Substitution Stability)
$[\Omega]\Delta \vdash \forall \alpha : \kappa.\, [\Omega, \alpha : \kappa]A_0 \leq^{-} [\Omega]B$	By Lemma 52 (Context Partitioning) + thinning
$[\Omega]\Delta \vdash \forall \alpha : \kappa.\, [\Omega]A_0 \leq^{-} [\Omega]B$	By def. of substitution
$[\Omega]\Delta \vdash [\Omega](\forall \alpha : \kappa.\, A_0) \leq^{-} [\Omega]B$	By def. of substitution

- **Case** $<:\exists R$: Similar to the $<:\forall L$ case.

- **Case** $\dfrac{\Gamma, \beta : \kappa \vdash A <:^{-} B_0 \dashv \Delta, \beta : \kappa, \Theta}{\Gamma \vdash A <:^{-} \forall \beta : \kappa.\, B_0 \dashv \Delta}\, <:\forall R$

$\Gamma, \beta : \kappa \vdash A <:^- B_0 \dashv \Delta, \beta : \kappa, \Theta$	Subderivation
Let $\Omega_Z = \lvert\Theta\rvert$.	
Let $\Omega' = (\Omega, \beta : \kappa, \Omega_Z)$.	
$(\Delta, \beta : \kappa, \Theta) \longrightarrow \Omega'$	By Lemma 25 (Filling Completes)

$\Gamma \vdash A$ *type*	Given
$\Gamma, \beta : \kappa \vdash A$ *type*	By Lemma 35 (Suffix Weakening)
$\Gamma \vdash \forall\beta : \kappa.\ B_0$ *type*	Given
$\Gamma, \beta : \kappa \vdash B_0$ *type*	By inversion (ForallWF)

$[\Omega'](\Delta, \beta : \kappa, \Theta) \vdash [\Omega']A \leq^- [\Omega']B_0$	By i.h.
$\Gamma, \beta : \kappa \longrightarrow \Delta, \beta : \kappa, \Theta$	By Lemma 50 (Subtyping Extension)
Θ is soft	By Lemma 22 (Extension Inversion) (i)
$[\Omega, \beta : \kappa](\Delta, \beta : \kappa) \vdash [\Omega, \beta : \kappa]A \leq^- [\Omega, \beta : \kappa]B_0$	By Lemma 17 (Substitution Stability)
$[\Omega, \beta : \kappa](\Delta, \beta : \kappa) \vdash [\Omega]A \leq^- [\Omega]B_0$	By def. of substitution
$[\Omega]\Delta \vdash [\Omega]A \leq^- \forall\beta : \kappa.\ [\Omega]B_0$	By $\leq\forall$R
$[\Omega]\Delta \vdash [\Omega]A \leq^- [\Omega](\forall\beta : \kappa.\ B_0)$	By def. of substitution

- **Case** <:∃L: Similar to the <:∀R case.

- **Case**
$$\frac{\Gamma \vdash A \equiv B \dashv \Delta}{\Gamma \vdash A <:^{\pm} B \dashv \Delta}\ <:\text{Equiv}$$

$\Gamma \vdash A \equiv B \dashv \Delta$	Subderivation
$\Delta \longrightarrow \Omega$	Given
$[\Omega]A = [\Omega]B$	By Lemma 86 (Soundness of Algorithmic Equivalence)
$\Gamma \longrightarrow \Delta$	By Lemma 49 (Equivalence Extension)
$\Gamma \vdash A$ *type*	Given
$[\Omega]\Omega \vdash [\Omega]A$ *type*	By Lemma 16 (Substitution for Type Well-Formedness)
$[\Omega]\Delta \vdash [\Omega]A$ *type*	By Lemma 54 (Completing Stability)
☞ $[\Omega]\Delta \vdash [\Omega]A \leq^{\pm} [\Omega]B$	By $\leq$Refl$\pm$

- **Case**
$$\frac{\Gamma \vdash A <:^- B \dashv \Delta \qquad \begin{array}{l} neg(A) \\ nonpos(B) \end{array}}{\Gamma \vdash A <:^+ B \dashv \Delta}\ <:^-_+\text{L}$$

$\Gamma \vdash A <:^- B \dashv \Delta$	By inversion
$neg(A)$	By inversion
$nonpos(B)$	By inversion
$nonpos(A)$	since $neg(A)$
$[\Omega]\Gamma \vdash [\Omega]A \leq^- [\Omega]B$	By induction
☞ $[\Omega]\Gamma \vdash [\Omega]A \leq^+ [\Omega]B$	By $\leq^-_+$

- **Case**
$$\frac{\Gamma \vdash A <:^- B \dashv \Delta \qquad \begin{array}{l} nonpos(A) \\ neg(B) \end{array}}{\Gamma \vdash A <:^+ B \dashv \Delta}\ <:^-_+\text{R}$$

  Similar to the $<:^-_+$L case.

- **Case**
$$\frac{\Gamma \vdash A <:^+ B \dashv \Delta \qquad \begin{array}{l} pos(A) \\ nonneg(B) \end{array}}{\Gamma \vdash A <:^- B \dashv \Delta}\ <:^+_-\text{L}$$

  Similar to the $<:^-_+$L case.

- **Case**
$$\frac{\Gamma \vdash A <:^{+} B \dashv \Delta \qquad \begin{array}{l} nonneg(A) \\ pos(B) \end{array}}{\Gamma \vdash A <:^{-} B \dashv \Delta} <:^{+}_{-}\mathsf{R}$$
Similar to the $<:^{-}_{+}\mathsf{L}$ case.

□

K′.4 Soundness of Typing

Theorem 7 (Soundness of Match Coverage)**.**

1. *If* $\Gamma \vdash \Pi$ *covers* $\vec{A}$ *and* $\Gamma \longrightarrow \Omega$ *and* $\Gamma \vdash \vec{A}$ **!** *types and* $[\Gamma]\vec{A} = \vec{A}$ *then* $[\Omega]\Gamma \vdash \Pi$ *covers* $\vec{A}$*.*
2. *If* $\Gamma \;/\; P \vdash \Pi$ *covers* $\vec{A}$ *and* $\Gamma \longrightarrow \Omega$ *and* $\Gamma \vdash \vec{A}$ **!** *types and* $[\Gamma]\vec{A} = \vec{A}$ *and* $[\Gamma]P = P$ *then* $[\Omega]\Gamma \;/\; P \vdash \Pi$ *covers* $\vec{A}$*.*

Proof. By mutual induction on the given algorithmic coverage derivation.

1. - **Case**
$$\frac{}{\Gamma \vdash \cdot \Rightarrow e_1 \mid \ldots \mathit{covers}\ \cdot}\ \mathsf{CoversEmpty}$$
$[\Omega]\Gamma \vdash \cdot \Rightarrow e_1 \mid \ldots \mathit{covers}\ \cdot$ By DeclCoversEmpty
   - **Cases** CoversVar, Covers1, Covers×, Covers+, Covers∃, Covers∧, CoversVec:
   Use the i.h. and apply the corresponding declarative rule.

2. - **Case**
$$\frac{\Gamma \;/\; [\Gamma]t_1 \doteq [\Gamma]t_2 : \kappa \dashv \Delta \qquad \Delta \vdash [\Delta]\Pi \mathit{\ covers\ } [\Delta]\vec{A}}{\Gamma \;/\; t_1 = t_2 \vdash \Pi \mathit{\ covers\ } \vec{A}}\ \mathsf{CoversEq}$$
$\Gamma \;/\; [\Gamma]t_1 \doteq [\Gamma]t_2 : \kappa \dashv \Delta$ Subderivation
$\Delta \vdash [\Delta]\Pi \mathit{\ covers\ } [\Delta]\vec{A}$ Subderivation
$[\Omega]\Delta \vdash [\Delta]\Pi \mathit{\ covers\ } [\Delta]A_0, [\Delta]\vec{A})$ By i.h.
$\Delta = (\Gamma, \Theta)$ By Lemma 88 (Soundness of Equality Elimination) (1)
$\mathsf{mgu}(t_1, t_2) = \theta$ ″
$\ldots$ ″
$[\Omega]\Delta = [\theta][\Omega]\Gamma$ By Lemma 93 (Substitution Upgrade) (iii)
$[\Delta]\Pi = [\theta]\Pi$ By Lemma 93 (Substitution Upgrade) (iv)
$([\Delta]\vec{A}) = ([\theta]A_0, [\theta]\vec{A})$ By Lemma 93 (Substitution Upgrade) (i)
$[\theta][\Omega]\Gamma \vdash [\theta]\Pi \mathit{\ covers\ } [\theta]\vec{A}$ By above equalities
☞ $[\Omega]\Gamma \;/\; t_1 = t_2 \vdash \Pi \mathit{\ covers\ } \vec{A}$ By DeclCoversEq

   - **Case**
$$\frac{\Gamma \;/\; [\Gamma]t_1 \doteq [\Gamma]t_2 : \kappa \dashv \bot}{\Gamma \;/\; t_1 = t_2 \vdash \Pi \mathit{\ covers\ } \vec{A}}\ \mathsf{CoversEqBot}$$
$\Gamma \;/\; [\Gamma]t_1 \doteq [\Gamma]t_2 : \kappa \dashv \bot$ Subderivation
$\mathsf{mgu}([\Gamma]t_1, [\Gamma]t_2) = \bot$ By Lemma 88 (Soundness of Equality Elimination) (2)
$\mathsf{mgu}(t_1, t_2) = \bot$ By given equality
☞ $[\Omega]\Gamma \;/\; t_1 = t_2 \vdash \Pi \mathit{\ covers\ } \vec{A}$ By DeclCoversEqBot □

Lemma 89 (Well-formedness of Algorithmic Typing)**.**
Given Γ *ctx:*

(i) If $\Gamma \vdash e \Rightarrow A\ p \dashv \Delta$ *then* $\Delta \vdash A\ p$ *type.*

(ii) If $\Gamma \vdash s : A\ p \gg B\ q \dashv \Delta$ *and* $\Gamma \vdash A\ p$ *type then* $\Delta \vdash B\ q$ *type.*

Proof. 1. Suppose $\Gamma \vdash e \Rightarrow A\ p \dashv \Delta$:

- **Case** $\dfrac{(x : A\ p) \in \Gamma}{\Gamma \vdash x \Rightarrow [\Gamma]A\ p \dashv \Gamma}\ \mathsf{Var}$

  $\Gamma = (\Gamma_0, x : A\ p, \Gamma_1) \quad (x : A\ p) \in \Gamma$
  $\Gamma \vdash A\ p$ *type* Follows from Γ *ctx*

- **Case** $\dfrac{\Gamma \vdash A\mathbf{!}\ \textit{type} \qquad \Gamma \vdash e \Leftarrow [\Gamma]A\ \mathbf{!} \dashv \Delta}{\Gamma \vdash (e : A) \Rightarrow [\Delta]A\ \mathbf{!} \dashv \Delta}\ \mathsf{Anno}$

  $\Gamma \vdash A\ \mathbf{!}$ *type* By inversion
  $\Gamma \longrightarrow \Delta$ By Lemma 51 (Typing Extension)
  $\Delta \vdash A\ \mathbf{!}$ *type* By Lemma 41 (Extension Weakening for Principal Typing)
  ☞ $\Delta \vdash [\Delta]A\ \mathbf{!}$ *type* By Lemma 39 (Principal Agreement) (i)

- **Case** $\dfrac{\Gamma \vdash e \Rightarrow A\ p \dashv \Theta \qquad \Theta \vdash s : [\Theta]A\ p \gg C\ q \dashv \Delta \qquad \begin{array}{l} p = \not{!} \text{ or } q = \mathbf{!} \\ \text{or } \mathsf{FEV}([\Delta]C) \neq \emptyset \end{array}}{\Gamma \vdash e\ s \Rightarrow C\ q \dashv \Delta}\ \rightarrow\mathsf{E}$

  $\Gamma \vdash e \Rightarrow A\ p \dashv \Theta$ By inversion
  $\Theta \vdash A\ p$ *type* By induction
  $\Theta \vdash [\Theta]A\ p$ *type* By Lemma 40 (Right-Hand Subst. for Principal Typing)
  Θ *ctx* By implicit assumption
  $\Theta \vdash s : [\Theta]A\ p \gg C\ q \dashv \Delta$ By inversion
  ☞ $\Delta \vdash C\ q$ *type* By mutual induction

- **Case** $\dfrac{\Gamma \vdash e \Rightarrow A\ \mathbf{!} \dashv \Theta \qquad \Theta \vdash s : [\Theta]A\ \mathbf{!} \gg C \dashv \Delta \qquad \mathsf{FEV}([\Delta]C) = \emptyset}{\Gamma \vdash e\ s \Rightarrow C\ \mathbf{!} \dashv \Delta}\ \rightarrow\text{E-!}$

  $\Gamma \vdash e \Rightarrow A\ p \dashv \Theta$ By inversion
  $\Theta \vdash A\ p$ *type* By induction
  $\Theta \vdash [\Theta]A\ p$ *type* By Lemma 40 (Right-Hand Subst. for Principal Typing)
  Θ *ctx* By implicit assumption
  $\Theta \vdash s : [\Theta]A\ p \gg C \dashv \Delta$ By inversion
  $\Delta \vdash C$ *type* By mutual induction
  $\mathsf{FEV}([\Delta]C) = \emptyset$ By inversion
  $\Delta \vdash C\mathbf{!}$ *type* By PrincipalWF

2. Suppose $\Gamma \vdash s : A\ p \gg B\ q \dashv \Delta$ and $\Gamma \vdash A\ p$ *type*:

- **Case** $\dfrac{}{\Gamma \vdash \cdot : A\ p \gg A\ p \dashv \Gamma}\ \mathsf{EmptySpine}$

  $\Gamma \vdash A\ p$ *type* Given

- **Case** $\dfrac{\Gamma \vdash e \Leftarrow A\ p \dashv \Theta \qquad \Theta \vdash s : [\Theta]B\ p \gg C\ q \dashv \Delta}{\Gamma \vdash e\ s : A \rightarrow B\ p \gg C\ q \dashv \Delta}\ \rightarrow\mathsf{Spine}$

  $\Gamma \vdash A \rightarrow B\ p$ *type* Given
  $\Gamma \vdash B\ p$ *type* By Lemma 42 (Inversion of Principal Typing)
  $\Theta \vdash B\ p$ *type* By Lemma 41 (Extension Weakening for Principal Typing)
  $\Theta \vdash [\Theta]B\ p$ *type* By Lemma 40 (Right-Hand Subst. for Principal Typing)
  $\Delta \vdash C\ q$ *type* By induction

- **Case** $\dfrac{\Gamma, \hat{\alpha} : \kappa \vdash\ e\ s : [\hat{\alpha}/\alpha]A\ \gg C\ q\ \dashv \Delta}{\Gamma \vdash\ e\ s : \forall \alpha : \kappa.\ A\ p \gg C\ q\ \dashv \Delta}\ \forall\mathsf{Spine}$

$\Gamma \vdash \forall \alpha : \kappa.\ A\ p\ \mathit{type}$	Given
$\Gamma \vdash \forall \alpha : \kappa.\ A\ \mathit{type}$	By inversion
$\Gamma, \alpha : \kappa \vdash A\ \mathit{type}$	By inversion
$\Gamma, \hat{\alpha} : \kappa, \alpha : \kappa \vdash A\ \mathit{type}$	By weakening
$\Gamma, \hat{\alpha} : \kappa \vdash [\hat{\alpha}/\alpha]A\ \mathit{type}$	By substitution
☞ $\Delta \vdash C\ q\ \mathit{type}$	By induction

- **Case** $\dfrac{\Gamma \vdash\ P\ \mathit{true} \dashv \Theta \qquad \Theta \vdash\ e\ s : [\Theta]A\ p \gg C\ q\ \dashv \Delta}{\Gamma \vdash\ e\ s : P \supset A\ p \gg C\ q\ \dashv \Delta}\ \supset\mathsf{Spine}$

$\Gamma \vdash P \supset A\ p\ \mathit{type}$	Given
$\Gamma \vdash P\ \mathit{prop}$	By Lemma 42 (Inversion of Principal Typing)
$\Gamma \vdash A\ p\ \mathit{type}$	$''$
$\Gamma \longrightarrow \Theta$	By Lemma 47 (Checkprop Extension)
$\Theta \vdash A\ p\ \mathit{type}$	By Lemma 41 (Extension Weakening for Principal Typing)
$\Theta \vdash [\Theta]A\ p\ \mathit{type}$	By Lemma 40 (Right-Hand Subst. for Principal Typing)
☞ $\Delta \vdash C\ q\ \mathit{type}$	By induction

- **Case** $\dfrac{\overbrace{\Gamma[\hat{\alpha}_2 : \star, \hat{\alpha}_1 : \star, \hat{\alpha} : \star = \hat{\alpha}_1 \to \hat{\alpha}_2]}^{\Theta} \vdash\ e\ s : (\hat{\alpha}_1 \to \hat{\alpha}_2)\ \gg C\ \dashv \Delta}{\Gamma[\hat{\alpha} : \star] \vdash\ e\ s : \hat{\alpha}\ \gg C\ \dashv \Delta}\ \hat{\alpha}\mathsf{Spine}$

$\Theta \vdash \hat{\alpha}_1 \to \hat{\alpha}_2\ \mathit{type}$	By rules
☞ $\Delta \vdash C\ q\ \mathit{type}$	By induction

□

Theorem 8 (Soundness of Algorithmic Typing)**.**
Given $\Delta \longrightarrow \Omega$*:*

(i) *If* $\Gamma \vdash\ e \Leftarrow A\ p \dashv \Delta$ *and* $\Gamma \vdash\ A\ p\ \mathit{type}$ *then* $[\Omega]\Delta \vdash\ [\Omega]e \Leftarrow [\Omega]A\ p$.

(ii) *If* $\Gamma \vdash\ e \Rightarrow A\ p \dashv \Delta$ *then* $[\Omega]\Delta \vdash\ [\Omega]e \Rightarrow [\Omega]A\ p$.

(iii) *If* $\Gamma \vdash\ s : A\ p \gg B\ q \dashv \Delta$ *and* $\Gamma \vdash\ A\ p\ \mathit{type}$ *then* $[\Omega]\Delta \vdash\ [\Omega]s : [\Omega]A\ p \gg [\Omega]B\ q$.

(iv) *If* $\Gamma \vdash\ s : A\ p \gg B\ \lceil q \rceil \dashv \Delta$ *and* $\Gamma \vdash\ A\ p\ \mathit{type}$ *then* $[\Omega]\Delta \vdash\ [\Omega]s : [\Omega]A\ p \gg [\Omega]B\ \lceil q \rceil$.

(v) *If* $\Gamma \vdash\ \Pi :: \vec{A} \Leftarrow C\ p \dashv \Delta$ *and* $\Gamma \vdash\ \vec{A}$ **!** *types and* $[\Gamma]\vec{A} = \vec{A}$ *and* $\Gamma \vdash\ C\ p\ \mathit{type}$
then $[\Omega]\Delta \vdash\ [\Omega]\Pi :: [\Omega]\vec{A} \Leftarrow [\Omega]C\ p$.

(vi) *If* $\Gamma\ /\ P \vdash\ \Pi :: \vec{A} \Leftarrow C\ p \dashv \Delta$ *and* $\Gamma \vdash\ P\ \mathit{prop}$ *and* $\mathsf{FEV}(P) = \emptyset$ *and* $[\Gamma]P = P$
and $\Gamma \vdash\ \vec{A}$ **!** *types and* $\Gamma \vdash\ C\ p\ \mathit{type}$
then $[\Omega]\Delta\ /\ [\Omega]P \vdash\ [\Omega]\Pi :: [\Omega]\vec{A} \Leftarrow [\Omega]C\ p$.

Proof. By induction, using the measure in Definition 7.

- **Case** $\dfrac{(x : A\ p) \in \Gamma}{\Gamma \vdash\ x \Rightarrow [\Gamma]A\ p \dashv \Gamma}\ \mathsf{Var}$

$(x : A\ p) \in \Gamma$	Premise
$(x : A\ p) \in \Delta$	$\Gamma = \Delta$
$\Delta \longrightarrow \Omega$	Given
$(x : [\Omega]A\ p) \in [\Omega]\Gamma$	By Lemma 9 (Uvar Preservation) (ii)
$[\Omega]\Gamma \vdash [\Omega]x \Rightarrow [\Omega]A\ p$	By DeclVar
$\Delta \longrightarrow \Omega$	Given
$\Gamma \longrightarrow \Omega$	$\Gamma = \Delta$
$[\Omega]A = [\Omega][\Gamma]A$	By Lemma 29 (Substitution Monotonicity) (iii)
☞ $[\Omega]\Gamma \vdash [\Omega]x \Rightarrow [\Omega][\Gamma]A\ p$	By above equality

- **Case** $\dfrac{\Gamma \vdash e \Rightarrow A\ q \dashv \Theta \qquad \Theta \vdash A <:^{\pm} B \dashv \Delta}{\Gamma \vdash e \Leftarrow B\ p \dashv \Delta}\ \mathsf{Sub}$

$\Gamma \vdash e \Rightarrow A\ q \dashv \Theta$	Subderivation
$\Theta \vdash A <:^{\pm} B \dashv \Delta$	Subderivation
$\Theta \longrightarrow \Delta$	By Lemma 51 (Typing Extension)
$\Delta \longrightarrow \Omega$	Given
$\Theta \longrightarrow \Omega$	By Lemma 33 (Extension Transitivity)
$[\Omega]\Theta \vdash [\Omega]e \Rightarrow [\Omega]A\ q$	By i.h.
$[\Omega]\Theta = [\Omega]\Delta$	By Lemma 56 (Confluence of Completeness)
$[\Omega]\Delta \vdash [\Omega]e \Rightarrow [\Omega]A\ q$	By above equality
$\Theta \vdash A <:^{\pm} B \dashv \Delta$	Subderivation
$[\Omega]\Delta \vdash [\Omega]A \leq^{\pm} [\Omega]B$	By Theorem 6 (Soundness of Algorithmic Subtyping)
☞ $[\Omega]\Delta \vdash [\Omega]e \Leftarrow [\Omega]B\ p$	By DeclSub

- **Case** $\dfrac{\Gamma \vdash A_0\mathbf{!}\ \mathit{type} \qquad \Gamma \vdash e_0 \Leftarrow [\Gamma]A_0\ \mathbf{!} \dashv \Delta}{\Gamma \vdash (e_0 : A_0) \Rightarrow [\Delta]A_0\ \mathbf{!} \dashv \Delta}\ \mathsf{Anno}$

$\Gamma \vdash e_0 \Leftarrow [\Gamma]A_0\ \mathbf{!} \dashv \Delta$	Subderivation
$[\Omega]\Delta \vdash [\Omega]e_0 \Leftarrow [\Omega][\Gamma]A_0\ \mathbf{!}$	By i.h.
$\Gamma \vdash A_0\mathbf{!}\ \mathit{type}$	Subderivation
$\Gamma \vdash A_0\ \mathit{type}$	By inversion
$\mathsf{FEV}(A_0) = \emptyset$	$''$
$\Gamma \longrightarrow \Delta$	By Lemma 51 (Typing Extension)
$\Delta \longrightarrow \Omega$	Given
$\Gamma \longrightarrow \Omega$	By Lemma 33 (Extension Transitivity)
$\Omega \vdash A_0\ \mathit{type}$	By Lemma 36 (Extension Weakening (Sorts))
$[\Omega]\Omega \vdash [\Omega]A_0\ \mathit{type}$	By Lemma 16 (Substitution for Type Well-Formedness)
$[\Omega]\Omega = [\Omega]\Delta$	By Lemma 54 (Completing Stability)
$[\Omega]\Delta \vdash [\Omega]A_0\ \mathit{type}$	By above equality
$[\Omega][\Gamma]A_0 = [\Omega]A_0$	By Lemma 29 (Substitution Monotonicity) (iii)
$[\Omega]\Delta \vdash [\Omega]e_0 \Leftarrow [\Omega]A_0\ \mathbf{!}$	By above equality
$[\Omega]\Delta \vdash ([\Omega]e_0 : [\Omega]A_0) \Rightarrow [\Omega]A_0\ \mathbf{!}$	By DeclAnno
$[\Omega]A_0 = A_0$	From definition of substitution
☞ $[\Omega]\Delta \vdash [\Omega](e_0 : A_0) \Rightarrow [\Omega]A_0\ \mathbf{!}$	By above equality

- **Case**

$$\frac{}{\Gamma \vdash ()\ \Leftarrow 1\ p \dashv \underbrace{\Gamma}_{\Delta}}\ 1\mathsf{I}$$

$[\Omega]\Delta \vdash ()\ \Leftarrow 1\ p$	By Decl1I
☞ $[\Omega]\Delta \vdash [\Omega]()\ \Leftarrow [\Omega]1\ p$	By definition of substitution

- **Case**

$$\frac{}{\Gamma_0[\hat{\alpha} : \star] \vdash ()\ \Leftarrow \hat{\alpha}\not{\mathbb{I}} \dashv \underbrace{\Gamma_0[\hat{\alpha} : \star = 1]}_{\Delta}}\ 1\mathsf{I}\hat{\alpha}$$

$\Gamma_0[\hat{\alpha} : \star = 1] \longrightarrow \Omega$	Given
$[\Omega]\hat{\alpha} = [\Omega][\Delta]\hat{\alpha}$	By Lemma 29 (Substitution Monotonicity) (i)
$= [\Omega]1$	By definition of context application
$= 1$	By definition of context application
$[\Omega]\Delta \vdash ()\ \Leftarrow 1\not{\mathbb{I}}$	By Decl1I
☞ $[\Omega]\Delta \vdash [\Omega]()\ \Leftarrow [\Omega]\hat{\alpha}\not{\mathbb{I}}$	By above equality

- **Case** $\dfrac{\nu\ \textit{chk-I} \qquad \Gamma, \alpha : \kappa \vdash \nu \Leftarrow A_0\ \mathsf{p} \dashv \Delta, \alpha : \kappa, \Theta}{\Gamma \vdash \nu \Leftarrow \forall \alpha : \kappa.\, A_0\ \mathsf{p} \dashv \Delta}\ \forall\mathsf{I}$

$$
\begin{array}{rll}
\Delta \longrightarrow \Omega & & \text{Given} \\
\Delta, \alpha \longrightarrow \Omega, \alpha & & \text{By } \longrightarrow\mathsf{Uvar} \\
\Gamma, \alpha \longrightarrow \Delta, \alpha, \Theta & & \text{By Lemma 51 (Typing Extension)} \\
\Theta\ \text{ soft} & & \text{By Lemma 22 (Extension Inversion) (i) (with } \Gamma_R = \cdot\text{, which is soft)} \\
\underbrace{\Delta, \alpha, \Theta}_{\Delta'} \longrightarrow \underbrace{\Omega, \alpha, |\Theta|}_{\Omega'} & & \text{By Lemma 25 (Filling Completes)} \\
\Gamma, \alpha \vdash \nu \Leftarrow A_0\ \mathsf{p} \dashv \Delta' & & \text{Subderivation} \\
\\
[\Omega']\Delta' \vdash [\Omega]\nu \Leftarrow [\Omega']A_0\ \mathsf{p} & & \text{By i.h.} \\
[\Omega']A_0 = [\Omega]A_0 & & \text{By Lemma 17 (Substitution Stability)} \\
[\Omega']\Delta' \vdash [\Omega]\nu \Leftarrow [\Omega]A_0\ \mathsf{p} & & \text{By above equality} \\
\\
\underbrace{\Delta, \alpha, \Theta}_{\Delta'} \longrightarrow \underbrace{\Omega, \alpha, |\Theta|}_{\Omega'} & & \text{Above} \\
\Theta \text{ is soft} & & \text{Above} \\
[\Omega']\Delta' = ([\Omega]\Delta, \alpha) & & \text{By Lemma 53 (Softness Goes Away)} \\
[\Omega]\Delta, \alpha \vdash [\Omega]\nu \Leftarrow [\Omega]A_0\ \mathsf{p} & & \text{By above equality} \\
\\
[\Omega]\Delta \vdash [\Omega]\nu \Leftarrow \forall \alpha.\, [\Omega]A_0\ \mathsf{p} & & \text{By Decl}\forall\mathsf{I} \\
\text{☞}\quad [\Omega]\Delta \vdash [\Omega]\nu \Leftarrow [\Omega](\forall \alpha.\, A_0)\ \mathsf{p} & & \text{By definition of substitution}
\end{array}
$$

- **Case** $\dfrac{\Gamma, \hat{\alpha} : \kappa \vdash e\ s_0 : [\hat{\alpha}/\alpha]A_0 \not\Vdash \gg C\ \mathsf{q} \dashv \Delta}{\Gamma \vdash e\ s_0 : \forall \alpha : \kappa.\, A_0\ \mathsf{p} \gg C\ \mathsf{q} \dashv \Delta}\ \forall\mathsf{Spine}$

$$
\begin{array}{rll}
\Gamma, \hat{\alpha} : \kappa \vdash e\ s_0 : [\hat{\alpha}/\alpha]A_0 \not\Vdash \gg C\ \mathsf{q} \dashv \Delta & & \text{Subderivation} \\
[\Omega]\Delta \vdash [\Omega](e\ s_0) : [\Omega][\hat{\alpha}/\alpha]A_0 \not\Vdash \gg [\Omega]C\ \mathsf{q} & & \text{By i.h.} \\
[\Omega]\Delta \vdash [\Omega](e\ s_0) : \left[[\Omega]\hat{\alpha}/\alpha\right][\Omega]A_0 \not\Vdash \gg [\Omega]C\ \mathsf{q} & & \text{By a property of substitution} \\
\Gamma, \hat{\alpha} : \kappa \vdash \hat{\alpha} : \kappa & & \text{By VarSort} \\
\Gamma, \hat{\alpha} : \kappa \longrightarrow \Delta & & \text{By Lemma 51 (Typing Extension)} \\
\Delta \vdash \hat{\alpha} : \kappa & & \text{By Lemma 36 (Extension Weakening (Sorts))} \\
\Delta \longrightarrow \Omega & & \text{Given} \\
[\Omega]\Delta \vdash [\Omega]\hat{\alpha} : \kappa & & \text{By Lemma 58 (Bundled Substitution for Sorting)} \\
\\
[\Omega]\Delta \vdash [\Omega](e\ s_0) : \forall \alpha : \kappa.\, [\Omega]A_0\ \mathsf{p} \gg [\Omega]C\ \mathsf{q} & & \text{By Decl}\forall\mathsf{Spine} \\
\text{☞}\quad [\Omega]\Delta \vdash [\Omega](e\ s_0) : [\Omega](\forall \alpha : \kappa.\, A_0)\ \mathsf{p} \gg [\Omega]C\ \mathsf{q} & & \text{By def. of subst.}
\end{array}
$$

- **Case** $\dfrac{e\ \textit{chk-I} \qquad \Gamma \vdash P\ \textit{true} \dashv \Theta \qquad \Theta \vdash e \Leftarrow [\Theta]A_0\ p \dashv \Delta}{\Gamma \vdash e \Leftarrow A_0 \wedge P\ p \dashv \Delta}\ \wedge\mathsf{I}$

$\Gamma \vdash P\ \textit{true} \dashv \Theta$	Subderivation
$\Delta \longrightarrow \Omega$	Given
$\Theta \longrightarrow \Delta$	By Lemma 51 (Typing Extension)
$\Theta \longrightarrow \Omega$	By Lemma 33 (Extension Transitivity)
$[\Omega]\Theta \vdash [\Omega]P\ \textit{true}$	By Lemma 87 (Soundness of Checkprop)
$[\Omega]\Delta \vdash [\Omega]P\ \textit{true}$	By Lemma 56 (Confluence of Completeness)
$\Theta \vdash e \Leftarrow [\Theta]A_0\ p \dashv \Delta$	Subderivation
$[\Omega]\Delta \vdash [\Omega]e \Leftarrow ([\Omega][\Theta]A_0)\ p$	By i.h.
$[\Omega]\Delta \vdash [\Omega]e \Leftarrow ([\Omega][\Theta]A_0) \wedge [\Omega]P\ p$	By Decl$\wedge$I
$[\Omega][\Theta]A_0 = [\Omega]A_0$	By Lemma 29 (Substitution Monotonicity) (iii)
$[\Omega]\Delta \vdash [\Omega]e \Leftarrow ([\Omega]A_0) \wedge [\Omega]P\ p$	By above equality
☞ $[\Omega]\Delta \vdash [\Omega]e \Leftarrow [\Omega](A_0 \wedge P)\ p$	By def. of substitution

- **Case** $\dfrac{\Gamma \vdash t = \mathsf{zero}\ \textit{true} \dashv \Delta}{\Gamma \vdash [] \Leftarrow (\mathsf{Vec}\ t\ A)\ p \dashv \Delta}\ \mathsf{Nil}$

$\Gamma \vdash t = \mathsf{zero}\ \textit{true} \dashv \Delta$	Subderivation
$\Delta \longrightarrow \Omega$	Given
$[\Omega]\Delta \vdash [\Omega](t = \mathsf{zero})\ \textit{true}$	By Lemma 87 (Soundness of Checkprop)
$[\Omega]\Delta \vdash [\Omega]t = \mathsf{zero}\ \textit{true}$	By def. of substitution
☞ $[\Omega]\Delta \vdash [\Omega]\,[] \Leftarrow (\mathsf{Vec}\ [\Omega]t\ [\Omega]A)\ p$	By DeclNil

- **Case** $\dfrac{\Gamma', \blacktriangleright_{\hat{\alpha}}, \hat{\alpha} : \mathbb{N} \vdash t = \mathsf{succ}(\hat{\alpha})\ \textit{true} \dashv \Gamma' \qquad \begin{array}{l}\Gamma' \vdash e_1 \Leftarrow [\Gamma']A_0\ p \dashv \Theta \\ \Theta \vdash e_2 \Leftarrow [\Theta](\mathsf{Vec}\ \hat{\alpha}\ A_0)\not\parallel \dashv \Delta, \blacktriangleright_{\hat{\alpha}}, \Delta'\end{array}}{\Gamma \vdash e_1 :: e_2 \Leftarrow (\mathsf{Vec}\ t\ A_0)\ p \dashv \Delta}\ \mathsf{Cons}$

$\Gamma', \blacktriangleright_{\hat{\alpha}}, \hat{\alpha} : \mathbb{N} \vdash t = \mathsf{succ}(\hat{\alpha})\ \textit{true} \dashv \Gamma'$	Subderivation
$\Delta \longrightarrow \Omega$	Given
$\Gamma' \longrightarrow \Theta$	By Lemma 51 (Typing Extension)
$\Theta \longrightarrow \Delta, \blacktriangleright_{\hat{\alpha}}, \Delta'$	By Lemma 51 (Typing Extension)
$\Delta, \blacktriangleright_{\hat{\alpha}}, \Delta' \longrightarrow \Omega'$	By Lemma 25 (Filling Completes)
$\Gamma' \longrightarrow \Omega'$	By Lemma 33 (Extension Transitivity)
$[\Omega']\Gamma' \vdash [\Omega'](t = \mathsf{succ}(\hat{\alpha}))\ \textit{true}$	By Lemma 87 (Soundness of Checkprop)
$[\Omega'](\Delta, \blacktriangleright_{\hat{\alpha}}, \Delta') \vdash [\Omega'](t = \mathsf{succ}(\hat{\alpha}))\ \textit{true}$	By Lemma 56 (Confluence of Completeness)
$[\Omega'](\Delta, \blacktriangleright_{\hat{\alpha}}, \Delta') \vdash [\Omega](t = \mathsf{succ}(\hat{\alpha}))\ \textit{true}$	By Lemma 17 (Substitution Stability)
$[\Omega]\Delta \vdash [\Omega](t = \mathsf{succ}(\hat{\alpha}))\ \textit{true}$	By Lemma 52 (Context Partitioning) + thinning
1 $[\Omega]\Delta \vdash ([\Omega]t) = \mathsf{succ}([\Omega]\hat{\alpha})\ \textit{true}$	By def. of substitution
$\Gamma' \vdash e_1 \Leftarrow [\Gamma']A_0\ p \dashv \Theta$	Subderivation
$[\Omega']\Theta \vdash [\Omega']e_1 \Leftarrow ([\Omega'][\Gamma']A_0)\ p$	By i.h.
$[\Omega'][\Gamma']A_0 = [\Omega']A_0$	By Lemma 29 (Substitution Monotonicity) (iii)
$[\Omega']\Theta \vdash [\Omega']e_1 \Leftarrow [\Omega']A_0\ p$	By above equality
2 $[\Omega]\Delta \vdash [\Omega]e_1 \Leftarrow [\Omega]A_0\ p$	Similar to above
$\Theta \vdash e_2 \Leftarrow [\Theta](\mathsf{Vec}\ \hat{\alpha}\ A_0)\not\parallel \dashv \Delta, \blacktriangleright_{\hat{\alpha}}, \Delta'$	Subderivation
$[\Omega'](\Delta, \blacktriangleright_{\hat{\alpha}}, \Delta') \vdash [\Omega']e_2 \Leftarrow [\Omega'][\Theta](\mathsf{Vec}\ \hat{\alpha}\ A_0)\not\parallel$	By i.h.
$[\Omega]\Delta \vdash [\Omega]e_2 \Leftarrow [\Omega](\mathsf{Vec}\ \hat{\alpha}\ A_0)\not\parallel$	Similar to above
3 $[\Omega]\Delta \vdash [\Omega]e_2 \Leftarrow (\mathsf{Vec}\ ([\Omega]\hat{\alpha})\ [\Omega]A_0)\ p$	By def. of substitution
$[\Omega]\Delta \vdash ([\Omega]e_1) :: [\Omega]e_2 \Leftarrow \mathsf{Vec}\ ([\Omega]t)\ [\Omega]A_0\ p$	By DeclCons (premises: 1, 2, 3)
☞ $[\Omega]\Delta \vdash [\Omega](e_1 :: e_2) \Leftarrow [\Omega](\mathsf{Vec}\ t\ A_0)\ p$	By def. of substitution

- **Case** $\dfrac{\nu\ \textit{chk-I} \qquad \Gamma, \blacktriangleright_P \ /\ P \dashv \Theta^+ \qquad \Theta^+ \vdash \nu \Leftarrow [\Theta^+]A_0\ \mathbf{!} \dashv \Delta, \blacktriangleright_P, \Delta'}{\Gamma \vdash \nu \Leftarrow P \supset A_0\ \mathbf{!} \dashv \Delta}\ \supset\mathsf{I}$

$$
\begin{array}{rl}
\Gamma \vdash A\ \mathbf{!}\ \textit{type} & \text{Given} \\
\mathsf{FEV}([\Gamma]A) = \emptyset & \text{By inversion on rule PrincipalWF} \\
\mathsf{FEV}([\Gamma]P) = \emptyset & A = (P \supset A_0) \\
\\
\Gamma, \blacktriangleright_P \ /\ P \dashv \Theta^+ & \text{Subderivation} \\
\Gamma, \blacktriangleright_P \ /\ \sigma \doteq t : \kappa \dashv \Theta^+ & \text{By inversion} \\
\mathsf{FEV}([\Gamma]\sigma) \cup \mathsf{FEV}([\Gamma]t) = \emptyset & \text{By } \mathsf{FEV}([\Gamma]P) = \emptyset \text{ above} \\
\\
\Theta^+ = (\Gamma, \blacktriangleright_P, \Theta) & \text{By Lemma 88 (Soundness of Equality Elimination)} \\
[\Omega', \Theta]t' = [\theta][\Gamma, \blacktriangleright_P]t' & \text{″ (for all } \Omega' \text{ extending } (\Gamma, \blacktriangleright_P) \text{ and } t' \text{ s.t. } \Omega' \vdash t' : \kappa') \\
\theta = \mathsf{mgu}(\sigma, t) & \text{″} \\
\\
\Delta \longrightarrow \Omega & \text{Given} \\
\Theta^+ \longrightarrow \Delta, \blacktriangleright_P, \Delta' & \text{By Lemma 51 (Typing Extension)} \\
\Gamma, \blacktriangleright_P, \Theta \longrightarrow \Delta, \blacktriangleright_P, \Delta' & \text{By above equalities} \\
\text{Let } \Omega^+ = (\Omega, \blacktriangleright_P, \Delta'). & \\
\Delta, \blacktriangleright_P, \Theta \longrightarrow \Omega, \blacktriangleright_P, \Delta' & \text{By repeated } \longrightarrow\mathsf{Eqn} \\
\Theta^+ \longrightarrow \Omega^+ & \text{By Lemma 33 (Extension Transitivity)} \\
\\
[\Omega', \Theta]B = [\theta][\Gamma, \blacktriangleright_P]B & \text{By Lemma 93 (Substitution Upgrade) (i)} \\
 & \quad \text{(for all } \Omega' \text{ extending } (\Gamma, \blacktriangleright_P \text{ and } B \text{ s.t. } \Omega' \vdash B : \kappa') \\
\Theta^+ \vdash \nu \Leftarrow [\Theta^+]A_0\ \mathbf{!} \dashv \Delta, \blacktriangleright_P, \Delta' & \text{Subderivation} \\
[\Omega^+](\Delta, \blacktriangleright_P, \Delta') \vdash [\Omega]\nu \Leftarrow [\Omega^+][\Theta^+]A_0\ \mathbf{!} & \text{By i.h.} \\
\\
\Gamma, \blacktriangleright_P, \Theta \longrightarrow \Omega, \blacktriangleright_P, \Delta' & \text{By Lemma 33 (Extension Transitivity)} \\
\Gamma \longrightarrow \Omega & \text{By Lemma 22 (Extension Inversion)} \\
[\Omega^+][\Theta^+]A_0 = [\Omega^+]A_0 & \text{By Lemma 29 (Substitution Monotonicity)} \\
\quad = [\theta][\Omega, \blacktriangleright_P]A_0 & \text{Above, with } (\Omega, \blacktriangleright_P) \text{ as } \Omega' \text{ and } A_0 \text{ as } B \\
\quad = [\theta][\Omega]A_0 & \text{By def. of substitution} \\
\\
[\Omega, \blacktriangleright_P, \Theta](\Delta, \blacktriangleright_P, \Delta') = [\theta][\Omega]\Delta & \text{By Lemma 93 (Substitution Upgrade) (iii)} \\
\\
[\theta][\Omega]\Delta \vdash [\Omega][\theta]\nu \Leftarrow [\theta][\Omega]A_0\ \mathbf{!} & \text{By above equalities} \\
\\
[\Omega^+](\Delta, \blacktriangleright_P, \Delta') \ /\ (\sigma = t) \vdash [\Omega]\nu \Leftarrow [\Omega]A_0\ \mathbf{!} & \text{By DeclCheckUnify} \\
[\Omega^+](\Delta, \blacktriangleright_P, \Delta') = [\Omega]\Delta & \text{From def. of context application} \\
[\Omega]\Delta \ /\ (\sigma = t) \vdash [\Omega]\nu \Leftarrow [\Omega]A_0\ \mathbf{!} & \text{By above equality} \\
[\Omega]\Delta \vdash [\Omega]\nu \Leftarrow (\sigma = t) \supset [\Omega]A_0\ \mathbf{!} & \text{By Decl}\supset\mathsf{I} \\
[\Omega]\Delta \vdash [\Omega]\nu \Leftarrow ([\Omega]\sigma = [\Omega]t) \supset [\Omega]A_0\ \mathbf{!} & \text{By FEV condition above}
\end{array}
$$

- **Case** $\dfrac{\nu\ \textit{chk-I} \qquad \Gamma, \blacktriangleright_P \ /\ P \dashv \bot}{\Gamma \vdash \nu \Leftarrow P \supset A_0\ \mathbf{!} \dashv \underbrace{\Gamma}_{\Delta}}\ \supset\mathsf{I}\bot$

$$
\begin{array}{rl}
\Gamma, \blacktriangleright_P \ /\ P \dashv \bot & \text{Subderivation} \\
\Gamma, \blacktriangleright_P \ /\ \sigma \doteq t : \kappa \dashv \bot & \text{By inversion} \\
P = (\sigma = t) & \text{″} \\
\mathsf{FEV}([\Gamma]\sigma) \cup \mathsf{FEV}([\Gamma]t) = \emptyset & \text{As in } \supset\mathsf{I} \text{ case (above)} \\
\mathsf{mgu}(\sigma, t) = \bot & \text{By Lemma 88 (Soundness of Equality Elimination)}
\end{array}
$$

$[\Omega]\Delta \,/\, (\sigma = t) \vdash [\Omega]v \Leftarrow [\Omega]A_0$! — By DeclCheck$\perp$
$[\Omega]\Delta \vdash [\Omega]v \Leftarrow (\sigma = t) \supset [\Omega]A_0$! — By Decl$\supset$I
$[\Omega]\Delta \vdash [\Omega]v \Leftarrow \big([\Omega](\sigma = t)\big) \supset [\Omega]A_0$! — By above FEV condition
☞ $[\Omega]\Delta \vdash [\Omega]v \Leftarrow [\Omega]\big(P \supset A_0\big)$! — By def. of subst.
Let $\Omega' = \Omega$.
☞ $\Omega \longrightarrow \Omega'$ — By Lemma 32 (Extension Reflexivity)
☞ $\Delta \longrightarrow \Omega'$ — Given

- **Case** $\dfrac{\Gamma \vdash P\ \mathit{true} \dashv \Theta \qquad \Theta \vdash e\ s_0 : [\Theta]A_0\ p \gg C\ q \dashv \Delta}{\Gamma \vdash e\ s_0 : P \supset A_0\ p \gg C\ q \dashv \Delta}\ \supset\mathsf{Spine}$

$\Theta \vdash e\ s_0 : [\Theta]A_0\ p \gg C\ q \dashv \Delta$ — Subderivation

$\Theta \longrightarrow \Delta$ — By Lemma 51 (Typing Extension)
$\Delta \longrightarrow \Omega$ — Given
$\Theta \longrightarrow \Omega$ — By Lemma 33 (Extension Transitivity)

$[\Omega]\Delta \vdash [\Omega](e\ s_0) : [\Omega][\Theta]A_0\ p \gg [\Omega]C\ q$ — By i.h.
$[\Omega][\Theta]A_0 = [\Omega]A_0$ — By Lemma 29 (Substitution Monotonicity) (iii)
$[\Omega]\Delta \vdash [\Omega](e\ s_0) : [\Omega]A_0\ p \gg [\Omega]C\ q$ — By above equality

$\Gamma \vdash P\ \mathit{true} \dashv \Theta$ — Subderivation
$[\Omega]\Theta \vdash [\Omega]P\ \mathit{true}$ — By Lemma 95 (Completeness of Checkprop)
$[\Omega]\Theta = [\Omega]\Delta$ — By Lemma 56 (Confluence of Completeness)
$[\Omega]\Delta \vdash [\Omega]P\ \mathit{true}$ — By above equality

$[\Omega]\Delta \vdash [\Omega](e\ s_0) : ([\Omega]P) \supset [\Omega]A_0\ p \gg [\Omega]C\ q$ — By Decl$\supset$Spine
☞ $[\Omega]\Delta \vdash [\Omega](e\ s_0) : [\Omega](P \supset A_0)\ p \gg [\Omega]C\ q$ — By def. of subst.

- **Case** $\dfrac{\Gamma, x : A_1\ p \vdash e_0 \Leftarrow A_2\ p \dashv \Delta, x : A_1\ p, \Theta}{\Gamma \vdash \lambda x.\, e_0 \Leftarrow A_1 \to A_2\ p \dashv \Delta}\ \to\mathsf{I}$

$\Delta \longrightarrow \Omega$ — Given
$\Delta, x : A_1\ p \longrightarrow \Omega, x : [\Omega]A_1\ p$ — By $\longrightarrow$Var
$\Gamma, x : A_1\ p \longrightarrow \Delta, x : A_1\ p, \Theta$ — By Lemma 51 (Typing Extension)
Θ soft — By Lemma 22 (Extension Inversion) (v) (with $\Gamma_R = \cdot$, which is soft)
$\underbrace{\Delta, x : A_1\ p, \Theta}_{\Delta'} \longrightarrow \underbrace{\Omega, x : [\Omega]A_1\ p, |\Theta|}_{\Omega'}$ — By Lemma 25 (Filling Completes)

$\Gamma, x : A_1\ p \vdash e_0 \Leftarrow A_2\ p \dashv \Delta'$ — Subderivation

$[\Omega']\Delta' \vdash [\Omega]e_0 \Leftarrow [\Omega']A_2\ p$ — By i.h.
$[\Omega']A_2 = [\Omega]A_2$ — By Lemma 17 (Substitution Stability)
$[\Omega']\Delta' \vdash [\Omega]e_0 \Leftarrow [\Omega]A_2\ p$ — By above equality

$\underbrace{\Delta, x : A_1\ p, \Theta}_{\Delta'} \longrightarrow \underbrace{\Omega, x : [\Omega]A_1\ p, |\Theta|}_{\Omega'}$ — Above
Θ soft — Above
$[\Omega']\Delta' = ([\Omega]\Delta, x : [\Omega]A_1\ p)$ — By Lemma 53 (Softness Goes Away)
$[\Omega]\Delta, x : [\Omega]A_1\ p \vdash [\Omega]e_0 \Leftarrow [\Omega]A_2\ p$ — By above equality
$[\Omega]\Delta \vdash \lambda x.\, [\Omega]e_0 \Leftarrow ([\Omega]A_1) \to ([\Omega]A_2)\ p$ — By Decl$\to$I
☞ $[\Omega]\Delta \vdash [\Omega](\lambda x.\, e_0) \Leftarrow [\Omega](A_1 \to A_2)\ p$ — By definition of substitution

- **Case** $\dfrac{v\ \mathit{chk\text{-}I} \qquad \Gamma, x : A\ p \vdash v \Leftarrow A\ p \dashv \Delta, x : A\ p, \Theta}{\Gamma \vdash \mathsf{rec}\ x.\, v \Leftarrow A\ p \dashv \Delta}\ \mathsf{Rec}$

Similar to the $\to$I case, applying DeclRec instead of Decl$\to$I.

- **Case** $\dfrac{\Gamma[\hat\alpha_1{:}\star, \hat\alpha_2{:}\star, \hat\alpha : \star = \hat\alpha_1 \to \hat\alpha_2], x : \hat\alpha_1 \not\Vdash \ e_0 \Leftarrow \hat\alpha_2 \not\Vdash \ \dashv \Delta, x : \hat\alpha_1 \not\Vdash, \Theta}{\Gamma[\hat\alpha : \star] \vdash \ \lambda x.\, e_0 \Leftarrow \hat\alpha \not\Vdash \ \dashv \Delta} \to\text{I}\hat\alpha$

$\Gamma[\hat\alpha_1{:}\star, \hat\alpha_2{:}\star, \hat\alpha : \star = \hat\alpha_1 \to \hat\alpha_2], x : \hat\alpha \not\Vdash \longrightarrow \Delta, x : \hat\alpha \not\Vdash, \Theta$ By Lemma 51 (Typing Extension)

Θ soft By Lemma 22 (Extension Inversion) (v) (with $\Gamma_R = \cdot$, which is soft)

$\Gamma[\hat\alpha_1{:}\star, \hat\alpha_2{:}\star, \hat\alpha : \star = \hat\alpha_1 \to \hat\alpha_2] \longrightarrow \Delta$ $''$

$\Delta \longrightarrow \Omega$ Given

$\Delta, x : \hat\alpha_1 \not\Vdash \longrightarrow \Omega, x : [\Omega]\hat\alpha_1 \not\Vdash$ By $\longrightarrow$Var

$\underbrace{\Delta, x : \hat\alpha_1 \not\Vdash, \Theta}_{\Delta'} \longrightarrow \underbrace{\Omega, x : [\Omega]\hat\alpha_1 \not\Vdash, |\Theta|}_{\Omega'}$ By Lemma 25 (Filling Completes)

$\Gamma[\hat\alpha_1{:}\star, \hat\alpha_2{:}\star, \hat\alpha : \star = \hat\alpha_1 \to \hat\alpha_2], x : \hat\alpha_1 \not\Vdash \vdash e_0 \Leftarrow \hat\alpha_2 \not\Vdash \dashv \Delta, x : \hat\alpha_1 \not\Vdash, \Theta$ Subderivation

$[\Omega']\Delta' \vdash [\Omega']e_0 \Leftarrow [\Omega']\hat\alpha_2 \not\Vdash$ By i.h.

$[\Omega']\hat\alpha_2 = \big[\Omega, x : [\Omega]\hat\alpha_1 \not\Vdash\big]\hat\alpha_2$ By Lemma 17 (Substitution Stability)

$= [\Omega]\hat\alpha_2$ By definition of substitution

$[\Omega']\Delta' = \big[\Omega, x : [\Omega]\hat\alpha_1 \not\Vdash\big](\Delta, x : \hat\alpha_1 \not\Vdash)$ By Lemma 53 (Softness Goes Away)

$= [\Omega]\Delta, x : [\Omega]\hat\alpha_1 \not\Vdash$ By definition of context substitution

$[\Omega]\Delta, x : [\Omega]\hat\alpha_1 \not\Vdash \vdash [\Omega]e_0 \Leftarrow [\Omega]\hat\alpha_2 \not\Vdash$ By above equalities

$[\Omega]\Delta \vdash \lambda x.\, [\Omega]e_0 \Leftarrow ([\Omega]\hat\alpha_1) \to [\Omega]\hat\alpha_2 \not\Vdash$ By Decl$\to$I

$\Gamma[\hat\alpha_1{:}\star, \hat\alpha_2{:}\star, \hat\alpha : \star = \hat\alpha_1 \to \hat\alpha_2] \longrightarrow \Omega$ Above and Lemma 33 (Extension Transitivity)

$[\Omega]\hat\alpha = [\Omega][\Gamma]\hat\alpha$ By Lemma 29 (Substitution Monotonicity) (i)

$= [\Omega]\big(([\Gamma]\hat\alpha_1) \to [\Gamma]\hat\alpha_2\big)$ By definition of substitution

$= ([\Omega][\Gamma]\hat\alpha_1) \to ([\Omega][\Gamma]\hat\alpha_2)$ By definition of substitution

$= ([\Omega]\hat\alpha_1) \to ([\Omega]\hat\alpha_2)$ By Lemma 29 (Substitution Monotonicity) (i)

☞ $[\Omega]\Delta \vdash [\Omega](\lambda x.\, e_0) \Leftarrow [\Omega]\hat\alpha \not\Vdash$ By above equality

- **Case** $\dfrac{\Gamma \vdash \ e_0 \Rightarrow A \ \mathsf{q} \ \dashv \Theta \qquad \Theta \vdash \ s_0 : A \ \mathsf{q} \gg C \ \lceil \mathsf{p} \rceil \ \dashv \Delta}{\Gamma \vdash \ e_0 \ s_0 \Rightarrow C \ \mathsf{p} \ \dashv \Delta} \to\text{E}$

$\Gamma \vdash \ e_0 \Rightarrow A \ \mathsf{q} \ \dashv \Theta$ Subderivation

$\Theta \vdash \ s_0 : A \ \mathsf{q} \gg C \ \lceil \mathsf{p} \rceil \ \dashv \Delta$ Subderivation

$\Gamma \longrightarrow \Theta$ and $\Theta \longrightarrow \Delta$ By Lemma 51 (Typing Extension)

$\Delta \longrightarrow \Omega$ Given

$\Theta \longrightarrow \Omega$ By Lemma 33 (Extension Transitivity)

$\Gamma \longrightarrow \Omega$ By Lemma 33 (Extension Transitivity)

$[\Omega]\Gamma = [\Omega]\Theta = [\Omega]\Delta$ By Lemma 56 (Confluence of Completeness)

$[\Omega]\Gamma \vdash [\Omega]e_0 \Rightarrow [\Omega]A \ \mathsf{q}$ By i.h.

$[\Omega]\Delta \vdash [\Omega]e_0 \Rightarrow [\Omega]A \ \mathsf{q}$ By above equality

$[\Omega]\Theta \vdash \ [\Omega]s_0 : [\Omega]A \ \mathsf{q} \gg [\Omega]C \ \lceil \mathsf{p} \rceil$ By i.h.

☞ $[\Omega]\Delta \vdash [\Omega](e_0 \ s_0) \Rightarrow [\Omega]C \ \mathsf{p}$ By rule Decl$\to$E

- **Case** $\dfrac{\Gamma \vdash \ s : A \ \mathbf{!} \gg C \not\Vdash \ \dashv \Delta \qquad \mathsf{FEV}(C) = \emptyset}{\Gamma \vdash \ s : A \ \mathbf{!} \gg C \ \lceil \mathbf{!} \rceil \ \dashv \Delta}$ SpineRecover

$\Gamma \vdash \ s : A \ \mathbf{!} \gg C \not\Vdash \ \dashv \Delta$ Subderivation

$[\Omega]\Gamma \vdash \ [\Omega]s : [\Omega]A \ \mathbf{!} \gg [\Omega]C \ \mathsf{q}$ By i.h.

We show the quantified premise of DeclSpineRecover, namely,

for all C'.
if $[\Omega]\Theta \vdash \ s : [\Omega]A \ \mathbf{!} \gg C' \not\Vdash$ then $C' = [\Omega]C$

Suppose we have C' such that $[\Omega]\Gamma \vdash\ s : [\Omega]A\ \mathbf{!} \gg C'\ \not\!\!\downarrow$. To apply DeclSpineRecover, we need to show $C' = [\Omega]C$.

$$
\begin{array}{rl}
[\Omega]\Gamma \vdash\ [\Omega]s : [\Omega]A\ \mathbf{!} \gg C'\ \not\!\!\downarrow & \text{Assumption} \\
\Omega_{canon} \longrightarrow \Omega & \text{By Lemma 59 (Canonical Completion)} \\
\mathsf{dom}(\Omega_{canon}) = \mathsf{dom}(\Gamma) & '' \\
\Gamma \longrightarrow \Omega_{canon} & '' \\
[\Omega]\Gamma = [\Omega_{canon}]\Gamma & \text{By Lemma 57 (Multiple Confluence)} \\
[\Omega]A = [\Omega_{canon}]A & \text{By Lemma 55 (Completing Completeness) (ii)} \\
[\Omega_{canon}]\Gamma \vdash\ [\Omega]s : [\Omega_{canon}]A\ \mathbf{!} \gg C'\ \not\!\!\downarrow & \text{By above equalities}
\end{array}
$$

$$
\begin{array}{rl}
\Gamma \vdash\ s : [\Gamma]A\ \mathbf{!} \gg C''\ q\ \dashv \Delta'' & \text{By Theorem 11 (Completeness of Algorithmic Typing)} \\
\Omega_{canon} \longrightarrow \Omega'' & '' \\
\Delta'' \longrightarrow \Omega'' & '' \\
C' = [\Omega'']C'' & ''
\end{array}
$$

$$
C'' = C \ \text{ and } \ q = \not\!\!\downarrow \ \text{ and } \ \Delta'' = \Delta \qquad \text{By Theorem 5 (Determinacy of Typing)}
$$

$$
\begin{array}{rl}
C' = [\Omega'']C'' & \text{Above} \\
= [\Omega'']C & \text{By above equality} \\
= [\Omega_{canon}]C & \text{By Lemma 55 (Completing Completeness) (ii)} \\
= [\Omega]C & \text{By Lemma 55 (Completing Completeness) (ii)}
\end{array}
$$

We have thus shown the above "for all C'. . . ." statement.

☞ $[\Omega]\Gamma \vdash\ [\Omega]s : [\Omega]A\ \mathbf{!} \gg [\Omega]C\ \lceil \mathbf{!} \rceil$ By DeclSpineRecover

- **Case** $\dfrac{\Gamma \vdash\ s : A\ p \gg C\ q\ \dashv \Delta \qquad \big((p = \not\!\!\downarrow)\ \text{ or }\ (q = \mathbf{!})\ \text{ or }\ (\mathsf{FEV}(C) \neq \emptyset)\big)}{\Gamma \vdash\ s : A\ p \gg C\ \lceil q \rceil\ \dashv \Delta}\ \mathsf{SpinePass}$

$$
\begin{array}{rll}
\Gamma \vdash\ s : A\ p \gg C\ q\ \dashv \Delta & & \text{Subderivation} \\
[\Omega]\Gamma \vdash\ [\Omega]s : [\Omega]A\ p \gg [\Omega]C\ q & & \text{By i.h.} \\
\text{☞}\quad [\Omega]\Gamma \vdash\ [\Omega]s : [\Omega]A\ p \gg [\Omega]C\ \lceil q \rceil & & \text{By DeclSpinePass}
\end{array}
$$

- **Case**

$$\frac{}{\Gamma \vdash\ \cdot : A\ p \gg A\ p\ \dashv \Gamma}\ \mathsf{EmptySpine}$$

☞ $[\Omega]\Gamma \vdash\ \cdot : [\Omega]A\ p \gg [\Omega]A\ p$ By DeclEmptySpine

- **Case** $\dfrac{\Gamma \vdash\ e_0 \Leftarrow A_1\ p\ \dashv \Theta \qquad \Theta \vdash\ s_0 : [\Theta]A_2\ p \gg C\ q\ \dashv \Delta}{\Gamma \vdash\ e_0\ s_0 : A_1 \to A_2\ p \gg C\ q\ \dashv \Delta}\ \to\mathsf{Spine}$

$$
\begin{array}{rll}
\Delta \longrightarrow \Omega & & \text{Given} \\
\Theta \longrightarrow \Delta & & \text{By Lemma 51 (Typing Extension)} \\
\Theta \longrightarrow \Omega & & \text{By Lemma 33 (Extension Transitivity)}
\end{array}
$$

$$
\begin{array}{rll}
\Gamma \vdash e_0 \Leftarrow A_1\ p\ \dashv \Theta & & \text{Subderivation} \\
[\Omega]\Theta \vdash [\Omega]e_0 \Leftarrow [\Omega]A_1\ p & & \text{By i.h.} \\
[\Omega]\Theta = [\Omega]\Delta & & \text{By Lemma 56 (Confluence of Completeness)} \\
[\Omega]\Delta \vdash [\Omega]e_0 \Leftarrow [\Omega]A_1\ p & & \text{By above equality} \\
\Theta \vdash\ s_0 : [\Theta]A_2\ p \gg C\ q\ \dashv \Delta & & \text{Subderivation} \\
[\Omega]\Delta \vdash\ [\Omega]s_0 : [\Omega][\Theta]A_2\ p \gg [\Omega]C\ q & & \text{By i.h.} \\
[\Omega][\Theta]A_2 = [\Omega]A_2 & & \text{By Lemma 29 (Substitution Monotonicity)} \\
[\Omega]\Delta \vdash\ [\Omega]s_0 : [\Omega]A_2\ p \gg [\Omega]C\ q & & \text{By above equality}
\end{array}
$$

$$
\begin{array}{rll}
[\Omega]\Delta \vdash\ [\Omega](e_0\ s_0) : ([\Omega]A_1) \to [\Omega]A_2\ p \gg [\Omega]C\ q & & \text{By Decl}\to\text{Spine} \\
\text{☞}\quad [\Omega]\Delta \vdash\ [\Omega](e_0\ s_0) : [\Omega](A_1 \to A_2)\ p \gg [\Omega]C\ q & & \text{By def. of subst.}
\end{array}
$$

- **Case** $\dfrac{\Gamma \vdash e_0 \Leftarrow A_k\ \mathsf{p} \dashv \Delta}{\Gamma \vdash \mathsf{inj}_k\, e_0 \Leftarrow A_1 + A_2\ \mathsf{p} \dashv \Delta}\ {+}\mathsf{I}_k$

$$\begin{array}{rll}
\Gamma \vdash e_0 \Leftarrow A_k\ \mathsf{p} \dashv \Delta & & \text{Subderivation} \\
[\Omega]\Delta \vdash [\Omega]e_0 \Leftarrow [\Omega]A_k\ \mathsf{p} & & \text{By i.h.} \\
[\Omega]\Delta \vdash \mathsf{inj}_k\,[\Omega]e_0 \Leftarrow ([\Omega]A_1) + ([\Omega]A_2)\ \mathsf{p} & & \text{By Decl}{+}\mathsf{I}_k \\
\text{☞}\quad [\Omega]\Delta \vdash [\Omega](\mathsf{inj}_k\, e_0) \Leftarrow [\Omega](A_1 + A_2)\ \mathsf{p} & & \text{By def. of substitution}
\end{array}$$

- **Case** $\dfrac{\Gamma[\hat\alpha_1 : \star, \hat\alpha_2 : \star, \hat\alpha : \star = \hat\alpha_1{+}\hat\alpha_2] \vdash e_0 \Leftarrow \hat\alpha_k \not\parallel \dashv \Delta}{\Gamma[\hat\alpha : \star] \vdash \mathsf{inj}_k\, e_0 \Leftarrow \hat\alpha \not\parallel \dashv \Delta}\ {+}\mathsf{I}\hat\alpha_k$

$$\begin{array}{rll}
\Gamma[\ldots, \hat\alpha : \star = \hat\alpha_1{+}\hat\alpha_2] \vdash e_0 \Leftarrow \hat\alpha_k \not\parallel \dashv \Delta & & \text{Subderivation} \\
[\Omega]\Delta \vdash [\Omega]e_0 \Leftarrow [\Omega]\hat\alpha_k \not\parallel & & \text{By i.h.} \\
[\Omega]\Delta \vdash \mathsf{inj}_k\,[\Omega]e_0 \Rightarrow ([\Omega]\hat\alpha_1) + ([\Omega]\hat\alpha_2) \not\parallel & & \text{By Decl}{+}\mathsf{I}_k \\
([\Omega]\hat\alpha_1) + ([\Omega]\hat\alpha_2) = [\Omega]\hat\alpha & & \text{Similar to the } {\to}\mathsf{I}\hat\alpha \text{ case (above)} \\
\text{☞}\quad [\Omega]\Delta \vdash [\Omega](\mathsf{inj}_k\, e_0) \Rightarrow [\Omega]\hat\alpha \not\parallel & & \text{By above equality / def. of subst.}
\end{array}$$

- **Case** $\dfrac{\Gamma \vdash e_1 \Leftarrow A_1\ \mathsf{p} \dashv \Theta \qquad \Theta \vdash e_2 \Leftarrow [\Theta]A_2\ \mathsf{p} \dashv \Delta}{\Gamma \vdash \langle e_1, e_2\rangle \Leftarrow A_1 \times A_2\ \mathsf{p} \dashv \Delta}\ {\times}\mathsf{I}$

$$\begin{array}{rll}
\Theta \vdash e_2 \Leftarrow [\Theta]A_2\ \mathsf{p} \dashv \Delta & & \text{Subderivation} \\
\Theta \longrightarrow \Delta & & \text{By Lemma 51 (Typing Extension)} \\
\Theta \longrightarrow \Omega & & \text{By Lemma 33 (Extension Transitivity)} \\
 & & \\
\Gamma \vdash e_1 \Leftarrow A_1\ \mathsf{p} \dashv \Theta & & \text{Subderivation} \\
[\Omega]\Theta \vdash [\Omega]e_1 \Leftarrow [\Omega]A_1\ \mathsf{p} & & \text{By i.h.} \\
[\Omega]\Delta \vdash [\Omega]e_1 \Leftarrow [\Omega]A_1\ \mathsf{p} & & \text{By Lemma 56 (Confluence of Completeness)} \\
 & & \\
\Theta \vdash e_2 \Leftarrow [\Theta]A_2\ \mathsf{p} \dashv \Delta & & \text{Subderivation} \\
[\Omega]\Delta \vdash [\Omega]e_2 \Leftarrow [\Omega][\Theta]A_2\ \mathsf{p} & & \text{By i.h.} \\
\Gamma \vdash A_1 \times A_2\ \mathit{type} & & \text{Given} \\
\Gamma \vdash A_2\ \mathit{type} & & \text{By inversion} \\
\Gamma \longrightarrow \Theta & & \text{By Lemma 51 (Typing Extension)} \\
\Theta \vdash A_2\ \mathit{type} & & \text{By Lemma 38 (Extension Weakening (Types))} \\
[\Omega]\Delta \vdash [\Omega]e_2 \Leftarrow [\Omega]A_2\ \mathsf{p} & & \text{By Lemma 29 (Substitution Monotonicity)} \\
 & & \\
[\Omega]\Delta \vdash \langle [\Omega]e_1, [\Omega]e_2\rangle \Leftarrow ([\Omega]A_1) \times [\Omega]A_2\ \mathsf{p} & & \text{By Decl}{\times}\mathsf{I} \\
\text{☞}\quad [\Omega]\Delta \vdash [\Omega]\langle e_1, e_2\rangle \Leftarrow [\Omega](A_1 \times A_2)\ \mathsf{p} & & \text{By def. of substitution}
\end{array}$$

- **Case** $\dfrac{\Gamma[\hat\alpha_2 : \star, \hat\alpha_1 : \star, \hat\alpha : \star = \hat\alpha_1{\times}\hat\alpha_2] \vdash e_1 \Leftarrow \hat\alpha_1 \not\parallel \dashv \Theta \qquad \Theta \vdash e_2 \Leftarrow [\Theta]\hat\alpha_2 \not\parallel \dashv \Delta}{\Gamma[\hat\alpha : \star] \vdash \langle e_1, e_2\rangle \Leftarrow \hat\alpha \not\parallel \dashv \Delta}\ {\times}\mathsf{I}\hat\alpha$

$$\begin{array}{rll}
\Delta \longrightarrow \Omega & & \text{Given} \\
\Theta \longrightarrow \Delta & & \text{By Lemma 51 (Typing Extension)} \\
\Theta \longrightarrow \Omega & & \text{By Lemma 33 (Extension Transitivity)} \\
\Gamma[\ldots, \hat\alpha : \star = \hat\alpha_1{\times}\hat\alpha_2] \vdash e_1 \Leftarrow \hat\alpha_1 \not\parallel \dashv \Theta & & \text{Subderivation} \\
[\Omega]\Theta \vdash [\Omega]e_1 \Leftarrow [\Omega]\hat\alpha_1 \not\parallel & & \text{By i.h.} \\
[\Omega]\Theta = [\Omega]\Delta & & \text{By Lemma 56 (Confluence of Completeness)} \\
[\Omega]\Delta \vdash [\Omega]e_1 \Leftarrow [\Omega]\hat\alpha_1 \not\parallel & & \text{By above equality} \\
 & & \\
\Theta \vdash e_2 \Leftarrow [\Theta]\hat\alpha_2 \not\parallel \dashv \Delta & & \text{Subderivation} \\
[\Omega]\Delta \vdash [\Omega]e_2 \Leftarrow [\Omega][\Theta]\hat\alpha_2 \not\parallel & & \text{By i.h.} \\
[\Omega][\Theta]\hat\alpha_2 = [\Omega]\hat\alpha_2 & & \text{By Lemma 29 (Substitution Monotonicity)} \\
[\Omega]\Delta \vdash [\Omega]e_2 \Leftarrow [\Omega]\hat\alpha_2 \not\parallel & & \text{By above equality} \\
[\Omega]\Delta \vdash \langle [\Omega]e_1, [\Omega]e_2\rangle \Leftarrow ([\Omega]\hat\alpha_1) \times [\Omega]\hat\alpha_2 \not\parallel & & \text{By Decl}{\times}\mathsf{I} \\
([\Omega]\hat\alpha_1) \times [\Omega]\hat\alpha_2 = [\Omega]\hat\alpha & & \text{Similar to the } {\to}\mathsf{I}\hat\alpha \text{ case (above)} \\
\text{☞}\quad [\Omega]\Delta \vdash [\Omega]\langle e_1, e_2\rangle \Leftarrow [\Omega]\hat\alpha \not\parallel & & \text{By above equality}
\end{array}$$

- **Case** $\dfrac{\Gamma[\hat{\alpha}_2 : \star, \hat{\alpha}_1 : \star, \hat{\alpha} : \star = \hat{\alpha}_1 \to \hat{\alpha}_2] \vdash e_0\ s_0 : (\hat{\alpha}_1 \to \hat{\alpha}_2) \not\!/\!/ \gg C \not\!/\!/ \dashv \Delta}{\Gamma[\hat{\alpha} : \star] \vdash e_0\ s_0 : \hat{\alpha} \not\!/\!/ \gg C \not\!/\!/ \dashv \Delta}\ \hat{\alpha}\mathsf{Spine}$

$\Gamma[\ldots, \hat{\alpha} : \star = \hat{\alpha}_1 \to \hat{\alpha}_2] \vdash e_0\ s_0 : (\hat{\alpha}_1 \to \hat{\alpha}_2) \not\!/\!/ \gg C \not\!/\!/ \dashv \Delta$	Subderivation
$[\Omega]\Delta \vdash [\Omega](e_0\ s_0) : [\Omega](\hat{\alpha}_1 \to \hat{\alpha}_2) \not\!/\!/ \gg [\Omega]C \not\!/\!/$	By i.h.
$[\Omega](\hat{\alpha}_1 \to \hat{\alpha}_2) = [\Omega]\hat{\alpha}$	Similar to the $\to$I$\hat{\alpha}$ case
☞ $[\Omega]\Delta \vdash [\Omega](e_0\ s_0) : [\Omega]\hat{\alpha} \not\!/\!/ \gg [\Omega]C \not\!/\!/$	By above equality

- **Case** $\dfrac{\Gamma \vdash e_0 \Rightarrow B\ \mathbf{!} \dashv \Theta \qquad \Theta \vdash \Pi :: [\Theta]B \Leftarrow [\Theta]C\ p \dashv \Delta \qquad \Delta \vdash \Pi\ \mathit{covers}\ [\Delta]B}{\Gamma \vdash \mathsf{case}(e_0, \Pi) \Leftarrow C\ p \dashv \Delta}\ \mathsf{Case}$

$\Gamma \vdash e_0 \Rightarrow B\ \mathbf{!} \dashv \Theta$	Subderivation
$\Theta \longrightarrow \Delta$	By Lemma 51 (Typing Extension)
$\Theta \longrightarrow \Omega$	By Lemma 33 (Extension Transitivity)
$[\Omega]\Theta \vdash [\Omega]e_0 \Rightarrow [\Omega]B\ \mathbf{!}$	By i.h.
$[\Omega]\Delta \vdash [\Omega]e_0 \Rightarrow [\Omega]B\ \mathbf{!}$	By Lemma 56 (Confluence of Completeness)

$\Theta \vdash \Pi :: [\Theta]B \Leftarrow [\Theta]C\ p \dashv \Delta$	Subderivation

$\Gamma \vdash e_0 \Rightarrow B\ \mathbf{!} \dashv \Theta$	Subderivation
$\Theta \vdash B\ \mathbf{!}\ \mathit{type}$	By Lemma 63 (Well-Formed Outputs of Typing) (Synthesis)

$\Gamma \vdash C\ p\ \mathit{type}$	Given
$\Gamma \longrightarrow \Theta$	By Lemma 51 (Typing Extension)
$\Theta \vdash C\ p\ \mathit{type}$	By Lemma 41 (Extension Weakening for Principal Typing)
$\Theta \vdash [\Theta]C\ p\ \mathit{type}$	By Lemma 40 (Right-Hand Subst. for Principal Typing)

$[\Omega]\Delta \vdash [\Omega]\Pi :: [\Omega]B \Leftarrow [\Omega][\Theta]C\ p$	By i.h. (v)
$[\Omega][\Theta]C = [\Omega]C$	By Lemma 29 (Substitution Monotonicity)
$[\Omega]\Delta \vdash [\Omega]\Pi :: [\Omega]B \Leftarrow [\Omega]C\ p$	By above equalities

$\Delta \vdash \Pi\ \mathit{covers}\ [\Delta]B$	Subderivation
$[\Delta][\Delta]B = [\Delta]B$	By idempotence of substitution
$\Theta \vdash B\ \mathbf{!}\ \mathit{type}$	By Lemma 63 (Well-Formed Outputs of Typing)
$\Delta \vdash B\ \mathbf{!}\ \mathit{type}$	By Lemma 41 (Extension Weakening for Principal Typing)
$\Delta \vdash [\Delta]B\ \mathbf{!}\ \mathit{type}$	By Lemma 40 (Right-Hand Subst. for Principal Typing)
$[\Omega]\Delta \vdash [\Omega]\Pi\ \mathit{covers}\ [\Delta]B$	By Theorem 7 (Soundness of Match Coverage)
$[\Delta]B = [\Omega]B$	By Lemma 39 (Principal Agreement) (i)
$[\Omega]\Delta \vdash [\Omega]\Pi\ \mathit{covers}\ [\Omega]B$	By above equality

☞ $[\Omega]\Delta \vdash [\Omega]\mathsf{case}(e_0, \Pi) \Leftarrow [\Omega]C\ p$	By DeclCase

Part (v):

- **Case** MatchEmpty: Apply rule DeclMatchEmpty.

- **Case** $\dfrac{\Gamma \vdash e \Leftarrow C\ p \dashv \Delta}{\Gamma \vdash (\cdot \Rightarrow e) :: \cdot \Leftarrow C\ p \dashv \Delta}\ \mathsf{MatchBase}$

  Apply the i.h. and DeclMatchBase.

- **Case** MatchUnit: Apply the i.h. and DeclMatchUnit.

- **Case** $\dfrac{\Gamma \vdash \pi :: \vec{A} \Leftarrow C\ p \dashv \Theta \qquad \Theta \vdash \Pi' :: \vec{A} \Leftarrow C\ p \dashv \Delta}{\Gamma \vdash \pi \mid \Pi' :: \vec{A} \Leftarrow C\ p \dashv \Delta}\ \mathsf{MatchSeq}$

  Apply the i.h. to each premise, using lemmas for well-formedness under Θ; then apply DeclMatchSeq.

- **Cases** Match$\exists$, Match$\wedge$, MatchWild, MatchNil, MatchCons:
  Apply the i.h. and the corresponding declarative match rule.
- **Cases** Match$\times$, Match$+_k$:
  We have $\Gamma \vdash \vec{A}$ **!** *types*, so the first type in $\vec{A}$ has no free existential variables.
  Apply the i.h. and the corresponding declarative match rule.
- **Case** $\dfrac{A \text{ not headed by } \wedge \text{ or } \exists \qquad \Gamma, z : A\,! \vdash \vec{\rho} \Rightarrow e' :: \vec{A} \Leftarrow C\ p \dashv \Delta, z : A\,!, \Delta'}{\Gamma \vdash z, \vec{\rho} \Rightarrow e :: A, \vec{A} \Leftarrow C\ p \dashv \Delta}$ MatchNeg

  Construct Ω' and show $\Delta, z : A\,!, \Delta' \longrightarrow \Omega'$ as in the $\to$I case.
  Use the i.h., then apply rule DeclMatchNeg.

Part (vi):

- **Case** $\dfrac{\Gamma \,/\, \sigma \doteq \tau : \kappa \dashv \bot}{\Gamma \,/\, \sigma = \tau \vdash \vec{\rho} p e :: \vec{A} \Leftarrow C\ p \dashv \Gamma}$ Match$\bot$

  $\Gamma \,/\, \sigma \doteq \tau : \kappa \dashv \bot$ Subderivation
  $[\Gamma](\sigma = \tau) = (\sigma = \tau)$ Given
  $(\sigma = \tau) = [\Gamma](\sigma = \tau)$ Given
  $= [\Omega](\sigma = \tau)$ By Lemma 29 (Substitution Monotonicity) (i)
  $\mathsf{mgu}(\sigma, \tau) = \bot$ By Lemma 88 (Soundness of Equality Elimination)
  $\mathsf{mgu}([\Omega]\sigma, [\Omega]\tau) = \bot$ By above equality
  ☞ $[\Omega]\Gamma \,/\, [\Omega](\sigma = \tau) \vdash [\Omega](\vec{\rho} p e) :: [\Omega]\vec{A} \Leftarrow [\Omega]C\ p$ By DeclMatch$\bot$
- **Case** $\dfrac{\Gamma, \blacktriangleright_p \,/\, \sigma \doteq \tau : \kappa \dashv \Gamma' \qquad \Gamma' \vdash \vec{\rho} \Rightarrow e :: \vec{A} \Leftarrow C\ p \dashv \Delta, \blacktriangleright_p, \Delta'}{\Gamma \,/\, \sigma = \tau \vdash \vec{\rho} \Rightarrow e :: \vec{A} \Leftarrow C\ p \dashv \Delta}$ MatchUnify

  $\Gamma, \blacktriangleright_p \,/\, \sigma \doteq \tau : \kappa \dashv \Gamma'$ Subderivation
  $(\sigma = \tau) = [\Gamma](\sigma = \tau)$ Given
  $= [\Omega](\sigma = \tau)$ By Lemma 29 (Substitution Monotonicity) (i)
  $\Gamma' = (\Gamma, \blacktriangleright_p, \Theta)$ By Lemma 88 (Soundness of Equality Elimination)
  $\Theta = ((\alpha_1 = t_1), \ldots, (\alpha_n = t_n))$ ″
  $\theta = \mathsf{mgu}([\Omega]\sigma, [\Omega]\tau)$ ″
  $[\Omega, \blacktriangleright_p, \Theta]t' = [\theta][\Omega, \blacktriangleright_p]t'$ ″ for all $\Omega, \blacktriangleright_p \vdash t' : \kappa'$
  $\Gamma, \blacktriangleright_p, \Theta \vdash \vec{\rho} \Rightarrow e :: \vec{A} \Leftarrow C\ p \dashv \Delta, \blacktriangleright_p, \Delta'$ Subderivation

  $[\Omega, \blacktriangleright_p, \Theta](\Delta, \blacktriangleright_p, \Delta') \vdash [\Omega, \blacktriangleright_p, \Theta](\vec{\rho} \Rightarrow e) :: [\Omega, \blacktriangleright_p, \Theta]\vec{A} \Leftarrow [\Omega, \blacktriangleright_p, \Theta]C\ p$ By i.h.
  $(\Omega, \blacktriangleright_p, \Theta) = [\theta](\Omega, \blacktriangleright_p)$ By Lemma 93 (Substitution Upgrade) (iii)
  $[\Omega, \blacktriangleright_p, \Theta]\vec{A} = [\theta][\Omega, \blacktriangleright_p]\vec{A}$ By Lemma 93 (Substitution Upgrade) (i)
  $[\Omega, \blacktriangleright_p, \Theta]C = [\theta][\Omega, \blacktriangleright_p]C$ By Lemma 93 (Substitution Upgrade) (i)
  $[\Omega, \blacktriangleright_p, \Theta](\vec{\rho} \Rightarrow e) = [\theta][\Omega](\vec{\rho} \Rightarrow e)$ By Lemma 93 (Substitution Upgrade) (iv)
  $\theta([\Omega, \blacktriangleright_p]\Gamma) \vdash [\theta][\Omega](\vec{\rho} \Rightarrow e) :: \theta([\Omega, \blacktriangleright_p]\vec{A}) \Leftarrow \theta([\Omega, \blacktriangleright_p]C)\ p$ By above equalities
  $\theta([\Omega]\Gamma) \vdash [\theta][\Omega](\vec{\rho} \Rightarrow e) :: \theta([\Omega]\vec{A}) \Leftarrow \theta([\Omega]C)\ p$ Subst. not affected by $\blacktriangleright_p$
  ☞ $[\Omega]\Gamma \,/\, [\Omega](\sigma = \tau) \vdash [\Omega](\vec{\rho} \Rightarrow e) :: [\Omega]\vec{A} \Leftarrow [\Omega]C\ p$ By DeclMatchUnify □

L′ Completeness

L′.1 Completeness of Auxiliary Judgments

Lemma 90 (Completeness of Instantiation)**.**
Given $\Gamma \longrightarrow \Omega$ *and* $\mathsf{dom}(\Gamma) = \mathsf{dom}(\Omega)$ *and* $\Gamma \vdash \tau : \kappa$ *and* $\tau = [\Gamma]\tau$ *and* $\hat{\alpha} \in \mathsf{unsolved}(\Gamma)$ *and* $\hat{\alpha} \notin \mathrm{FV}(\tau)$*:*
If $[\Omega]\hat{\alpha} = [\Omega]\tau$
then there are Δ, Ω' *such that* $\Omega \longrightarrow \Omega'$ *and* $\Delta \longrightarrow \Omega'$ *and* $\mathsf{dom}(\Delta) = \mathsf{dom}(\Omega')$ *and* $\Gamma \vdash \hat{\alpha} := \tau : \kappa \dashv \Delta$*.*

Proof. By induction on τ.
We have $[\Omega]\Gamma \vdash [\Omega]\hat{\alpha} \leq^* [\Omega]A$. We now case-analyze the shape of τ.

- **Case** $\tau = \hat{\beta}$:

	$\hat{\alpha} \notin \mathrm{FV}(\hat{\beta})$	Given
	$\hat{\alpha} \neq \hat{\beta}$	From definition of $\mathrm{FV}(-)$
	$\hat{\beta} \in \mathsf{unsolved}(\Gamma)$	From $[\Gamma]\hat{\beta} = \hat{\beta}$
Let $\Omega' = \Omega$.		
☞	$\Omega \longrightarrow \Omega'$	By Lemma 32 (Extension Reflexivity)

Now consider whether $\hat{\alpha}$ is declared to the left of $\hat{\beta}$, or vice versa.

  - **Case** $\Gamma = \Gamma_0[\hat{\alpha} : \kappa][\hat{\beta} : \kappa]$:

Let $\Delta = \Gamma_0[\hat{\alpha} : \kappa][\hat{\beta} : \kappa = \hat{\alpha}]$.		
	$\Gamma \vdash \hat{\alpha} := \hat{\beta} : \kappa \dashv \Delta$	By InstReach
	$[\Omega]\hat{\alpha} = [\Omega]\hat{\beta}$	Given
	$\Gamma \longrightarrow \Omega$	Given
☞	$\Delta \longrightarrow \Omega$	By Lemma 27 (Parallel Extension Solution)
☞	$\mathrm{dom}(\Delta) = \mathrm{dom}(\Omega')$	$\mathrm{dom}(\Delta) = \mathrm{dom}(\Gamma)$ and $\mathrm{dom}(\Omega') = \mathrm{dom}(\Omega)$

  - **Case** ($\Gamma = \Gamma_0[\hat{\beta} : \kappa][\hat{\alpha} : \kappa]$:
    Similar, but using InstSolve instead of InstReach.

- **Case** $\tau = \alpha$:

  We have $[\Omega]\hat{\alpha} = \alpha$, so (since Ω is well-formed), α is declared to the left of $\hat{\alpha}$ in Ω.
  We have $\Gamma \longrightarrow \Omega$.
  By Lemma 21 (Reverse Declaration Order Preservation), we know that α is declared to the left of $\hat{\alpha}$ in Γ; that is, $\Gamma = \Gamma_L[\alpha : \kappa][\hat{\alpha} : \kappa]$.
  Let $\Delta = \Gamma_L[\alpha : \kappa][\hat{\alpha} : \kappa = \alpha]$ and $\Omega' = \Omega$.
  By InstSolve, $\Gamma_L[\alpha : \kappa][\hat{\alpha} : \kappa] \vdash \hat{\alpha} := \alpha : \kappa \dashv \Delta$.
  By Lemma 27 (Parallel Extension Solution), $\Gamma_L[\alpha : \kappa][\hat{\alpha} : \kappa = \alpha] \longrightarrow \Omega$.
  We have $\mathrm{dom}(\Delta) = \mathrm{dom}(\Gamma)$ and $\mathrm{dom}(\Omega') = \mathrm{dom}(\Omega)$; therefore, $\mathrm{dom}(\Delta) = \mathrm{dom}(\Omega')$.

- **Case** $\tau = 1$:

  Similar to the $\tau = \alpha$ case, but without having to reason about where α is declared.

- **Case** $\tau = \mathsf{zero}$:

  Similar to the $\tau = 1$ case.

- **Case** $\tau = \tau_1 \oplus \tau_2$:

$[\Omega]\hat{\alpha} = [\Omega](\tau_1 \oplus \tau_2)$	Given
$= ([\Omega]\tau_1) \oplus ([\Omega]\tau_2)$	By definition of substitution
$\tau_1 \oplus \tau_2 = [\Gamma](\tau_1 \oplus \tau_2)$	Given
$\tau_1 = [\Gamma]\tau_1$	By definition of substitution and congruence
$\tau_2 = [\Gamma]\tau_2$	Similarly
$\hat{\alpha} \notin \mathrm{FV}(\tau_1 \oplus \tau_2)$	Given
$\hat{\alpha} \notin \mathrm{FV}(\tau_1)$	From definition of $\mathrm{FV}(-)$
$\hat{\alpha} \notin \mathrm{FV}(\tau_2)$	Similarly
$\Gamma = \Gamma_0[\hat{\alpha} : \star]$	By $\hat{\alpha} \in \mathsf{unsolved}(\Gamma)$
$\Gamma \longrightarrow \Omega$	Given
$\Gamma_0[\hat{\alpha} : \star] \longrightarrow \Gamma_0[\hat{\alpha}_2 : \star, \hat{\alpha}_1 : \star, \hat{\alpha} : \star]$	By Lemma 23 (Deep Evar Introduction) (i) twice
$\ldots, \hat{\alpha}_2, \hat{\alpha}_1 \vdash \hat{\alpha}_1 \oplus \hat{\alpha}_2 : \star$	Straightforward
$\Gamma_0[\hat{\alpha}_2, \hat{\alpha}_1, \hat{\alpha}] \longrightarrow \Gamma_0[\hat{\alpha}_2, \hat{\alpha}_1, \hat{\alpha} = \hat{\alpha}_1 \oplus \hat{\alpha}_2]$	By Lemma 23 (Deep Evar Introduction) (ii)
$\Gamma_0[\hat{\alpha}] \longrightarrow \Gamma_0[\hat{\alpha}_2, \hat{\alpha}_1, \hat{\alpha} = \hat{\alpha}_1 \oplus \hat{\alpha}_2]$	By Lemma 33 (Extension Transitivity)

(In the last few lines above, and the rest of this case, we omit the ": $\star$" annotations in contexts.)

Since $\hat{\alpha} \in \mathsf{unsolved}(\Gamma)$ and $\Gamma \longrightarrow \Omega$, we know that Ω has the form $\Omega_0[\hat{\alpha} = \tau_0]$.

To show that we can extend this context, we apply Lemma 23 (Deep Evar Introduction) (iii) twice to introduce $\hat{\alpha}_2 = \tau_2$ and $\hat{\alpha}_1 = \tau_1$, and then Lemma 28 (Parallel Variable Update) to overwrite τ_0:

$$\underbrace{\Omega_0[\hat{\alpha} = \tau_0]}_{\Omega} \longrightarrow \Omega_0[\hat{\alpha}_2 = \tau_2, \hat{\alpha}_1 = \tau_1, \hat{\alpha} = \hat{\alpha}_1 \oplus \hat{\alpha}_2]$$

We have $\Gamma \longrightarrow \Omega$, that is,

$$\Gamma_0[\hat{\alpha}] \longrightarrow \Omega_0[\hat{\alpha} = \tau_0]$$

By Lemma 26 (Parallel Admissibility) (i) twice, inserting unsolved variables $\hat{\alpha}_2$ and $\hat{\alpha}_1$ on both contexts in the above extension preserves extension:

$$\begin{array}{rcl} \Gamma_0[\hat{\alpha}_2, \hat{\alpha}_1, \hat{\alpha}] & \longrightarrow & \Omega_0[\hat{\alpha}_2 = \tau_2, \hat{\alpha}_1 = \tau_1, \hat{\alpha} = \tau_0] \quad \text{By Lemma 26 (Parallel Admissibility) (ii) twice} \\ \underbrace{\Gamma_0[\hat{\alpha}_2, \hat{\alpha}_1, \hat{\alpha} = \hat{\alpha}_1 \oplus \hat{\alpha}_2]}_{\Gamma_1} & \longrightarrow & \underbrace{\Omega_0[\hat{\alpha}_2 = \tau_2, \hat{\alpha}_1 = \tau_1, \hat{\alpha} = \hat{\alpha}_1 \oplus \hat{\alpha}_2]}_{\Omega_1} \quad \text{By Lemma 28 (Parallel Variable Update)} \end{array}$$

Since $\hat{\alpha} \notin \mathrm{FV}(\tau)$, it follows that $[\Gamma_1]\tau = [\Gamma]\tau = \tau$.
Therefore $\hat{\alpha}_1 \notin \mathrm{FV}(\tau_1)$ and $\hat{\alpha}_1, \hat{\alpha}_2 \notin \mathrm{FV}(\tau_2)$.
By Lemma 55 (Completing Completeness) (i) and (iii), $[\Omega_1]\Gamma_1 = [\Omega]\Gamma$ and $[\Omega_1]\hat{\alpha}_1 = \tau_1$.
By i.h., there are Δ_2 and Ω_2 such that $\Gamma_1 \vdash \hat{\alpha}_1 := \tau_1 : \kappa \dashv \Delta_2$ and $\Delta_2 \longrightarrow \Omega_2$ and $\Omega_1 \longrightarrow \Omega_2$.

Next, note that $[\Delta_2][\Delta_2]\tau_2 = [\Delta_2]\tau_2$.
By Lemma 64 (Left Unsolvedness Preservation), we know that $\hat{\alpha}_2 \in \mathsf{unsolved}(\Delta_2)$.
By Lemma 65 (Left Free Variable Preservation), we know that $\hat{\alpha}_2 \notin \mathrm{FV}([\Delta_2]\tau_2)$.
By Lemma 33 (Extension Transitivity), $\Omega \longrightarrow \Omega_2$.
We know $[\Omega_2]\Delta_2 = [\Omega]\Gamma$ because:

$$\begin{array}{rcll} [\Omega_2]\Delta_2 & = & [\Omega_2]\Omega_2 & \text{By Lemma 54 (Completing Stability)} \\ & = & [\Omega]\Omega & \text{By Lemma 55 (Completing Completeness) (iii)} \\ & = & [\Omega]\Gamma & \text{By Lemma 54 (Completing Stability)} \end{array}$$

By Lemma 55 (Completing Completeness) (i), we know that $[\Omega_2]\hat{\alpha}_2 = [\Omega_1]\hat{\alpha}_2 = \tau_2$.
By Lemma 55 (Completing Completeness) (i), we know that $[\Omega_2]\tau_2 = [\Omega]\tau_2$.
Hence we know that $[\Omega_2]\Delta_2 \vdash [\Omega_2]\hat{\alpha}_2 \leq^* [\Omega_2]\tau_2$.
By i.h., we have Δ and Ω' such that $\Delta_2 \vdash \hat{\alpha}_2 := [\Delta_2]\tau_2 : \kappa \dashv \Delta$ and $\Omega_2 \longrightarrow \Omega'$ and $\Delta \longrightarrow \Omega'$.
By rule InstBin, $\Gamma \vdash \hat{\alpha} := \tau : \kappa \dashv \Delta$.
By Lemma 33 (Extension Transitivity), $\Omega \longrightarrow \Omega'$.

- **Case** $\tau = \mathsf{succ}(\tau_0)$:

  Similar to the $\tau = \tau_1 \oplus \tau_2$ case, but simpler. □

Lemma 91 (Completeness of Checkeq)**.**
Given $\Gamma \longrightarrow \Omega$ *and* $\mathsf{dom}(\Gamma) = \mathsf{dom}(\Omega)$
and $\Gamma \vdash \sigma : \kappa$ *and* $\Gamma \vdash \tau : \kappa$
and $[\Omega]\sigma = [\Omega]\tau$
then $\Gamma \vdash [\Gamma]\sigma \triangleq [\Gamma]\tau : \kappa \dashv \Delta$
where $\Delta \longrightarrow \Omega'$ *and* $\mathsf{dom}(\Delta) = \mathsf{dom}(\Omega')$ *and* $\Omega \longrightarrow \Omega'$.

Proof. By mutual induction on the sizes of $[\Gamma]\sigma$ and $[\Gamma]\tau$.
We distinguish cases of $[\Gamma]\sigma$ and $[\Gamma]\tau$.

- **Case** $[\Gamma]\sigma = [\Gamma]\tau = 1$:

  ☞ $\Gamma \vdash 1 \triangleq 1 : \star \dashv \underbrace{\Gamma}_{\Delta}$ By CheckeqUnit

  Let $\Omega' = \Omega$.

  $\Gamma \longrightarrow \Omega$ Given

  ☞ $\Delta \longrightarrow \Omega'$ $\Delta = \Gamma$ and $\Omega' = \Omega$

  ☞ $\mathsf{dom}(\Gamma) = \mathsf{dom}(\Omega)$ Given

  ☞ $\Omega \longrightarrow \Omega'$ By Lemma 32 (Extension Reflexivity)

- **Case** $[\Gamma]\sigma = [\Gamma]t = \mathsf{zero}$:
  Similar to the case for 1, applying CheckeqZero instead of CheckeqUnit.
- **Case** $[\Gamma]\sigma = [\Gamma]t = \alpha$:
  Similar to the case for 1, applying CheckeqVar instead of CheckeqUnit.
- **Case** $[\Gamma]\sigma = \hat{\alpha}$ and $[\Gamma]t = \hat{\beta}$:
  - If $\hat{\alpha} = \hat{\beta}$: Similar to the case for 1, applying CheckeqVar instead of CheckeqUnit.
  - If $\hat{\alpha} \neq \hat{\beta}$:

	$\Gamma \longrightarrow \Omega$	Given
	$\hat{\alpha} \notin \mathrm{FV}(\underbrace{\hat{\beta}}_{[\Gamma]t})$	By definition of $\mathrm{FV}(-)$
	$[\Omega]\sigma = [\Omega]t$	Given
	$[\Omega][\Gamma]\sigma = [\Omega][\Gamma]t$	By Lemma 29 (Substitution Monotonicity) (i) twice
	$[\Omega]\hat{\alpha} = [\Omega][\Gamma]t$	$[\Gamma]\sigma = \hat{\alpha}$
	$\mathsf{dom}(\Gamma) = \mathsf{dom}(\Omega)$	Given
	$\Gamma \vdash \hat{\alpha} := [\Gamma]t : \kappa \dashv \Delta$	By Lemma 90 (Completeness of Instantiation)
☞	$\Omega \longrightarrow \Omega'$	$''$
☞	$\Delta \longrightarrow \Omega$	$''$
☞	$\mathsf{dom}(\Delta) = \mathsf{dom}(\Omega')$	$''$
☞	$\Gamma \vdash \hat{\alpha} \doteq [\Gamma]t : \kappa \dashv \Delta$	By CheckeqInstL

- **Case** $[\Gamma]\sigma = \hat{\alpha}$ and $[\Gamma]t = 1$ or zero or α:
  Similar to the previous case, except:
  $\hat{\alpha} \notin \mathrm{FV}(\underbrace{1}_{[\Gamma]t})$ By definition of $\mathrm{FV}(-)$
  and similarly for 1 and α.
- **Case** $[\Gamma]t = \hat{\alpha}$ and $[\Gamma]\sigma = 1$ or zero or α: Symmetric to the previous case.
- **Case** $[\Gamma]\sigma = \hat{\alpha}$ and $[\Gamma]t = \mathsf{succ}([\Gamma]t_0)$:
  If $\hat{\alpha} \notin \mathrm{FV}([\Gamma]t_0)$, then $\hat{\alpha} \notin \mathrm{FV}([\Gamma]t)$. Proceed as in the previous several cases.
  The other case, $\hat{\alpha} \in \mathrm{FV}([\Gamma]t_0)$, is impossible:
  We have $\hat{\alpha} \preceq [\Gamma]t_0$.
  Therefore $\hat{\alpha} \prec \mathsf{succ}([\Gamma]t_0)$, that is, $\hat{\alpha} \prec [\Gamma]t$.
  By a property of substitutions, $[\Omega]\hat{\alpha} \prec [\Omega][\Gamma]t$.
  Since $\Gamma \longrightarrow \Omega$, by Lemma 29 (Substitution Monotonicity) (i), $[\Omega][\Gamma]t = [\Omega]t$, so $[\Omega]\hat{\alpha} \prec [\Omega]t$.
  But it is given that $[\Omega]\hat{\alpha} = [\Omega]t$, a contradiction.
- **Case** $[\Gamma]t = \hat{\alpha}$ and $[\Gamma]\sigma = \mathsf{succ}([\Gamma]\sigma_0)$: Symmetric to the previous case.
- **Case** $[\Gamma]\sigma = [\Gamma]\sigma_1 \oplus [\Gamma]\sigma_2$ and $[\Gamma]t = [\Gamma]t_1 \oplus [\Gamma]t_2$:

	$\Gamma \longrightarrow \Omega$	Given
	$\Gamma \vdash [\Gamma]\sigma_1 \doteq [\Gamma]t_1 : \star \dashv \Theta$	By i.h.
	$\Theta \longrightarrow \Omega_0$	$''$
	$\Omega \longrightarrow \Omega_0$	$''$
	$\mathsf{dom}(\Theta) = \mathsf{dom}(\Omega_0)$	$''$
	$\Theta \vdash [\Theta][\Gamma]\sigma_2 \doteq [\Theta][\Gamma]t_2 : \star \dashv \Delta$	By i.h.
☞	$\Delta \longrightarrow \Omega'$	$''$
	$\Omega_0 \longrightarrow \Omega'$	$''$
☞	$\mathsf{dom}(\Delta) = \mathsf{dom}(\Omega')$	$''$
☞	$\Omega \longrightarrow \Omega'$	By Lemma 33 (Extension Transitivity)
☞	$\Gamma \vdash [\Gamma]\sigma_1 \oplus [\Gamma]\sigma_2 \doteq [\Gamma]t_1 \oplus [\Gamma]t_2) : \star \dashv \Delta$	By CheckeqBin

- **Case** $[\Gamma]\sigma = \hat{\alpha}$ and $[\Gamma]t = t_1 \oplus t_2$: Similar to the $\hat{\alpha}/\mathsf{succ}(-)$ case, showing the impossibility of $\hat{\alpha} \in \mathsf{FV}([\Gamma]t_k)$ for $k = 1$ and $k = 2$.
- **Case** $[\Gamma]t = \hat{\alpha}$ and $[\Gamma]\sigma = \sigma_1 \oplus \sigma_2$: Symmetric to the previous case. □

Lemma 92 (Completeness of Elimeq)**.**
If $[\Gamma]\sigma = \sigma$ *and* $[\Gamma]t = t$ *and* $\Gamma \vdash \sigma : \kappa$ *and* $\Gamma \vdash t : \kappa$ *and* $\mathsf{FEV}(\sigma) \cup \mathsf{FEV}(t) = \emptyset$ *then:*

(1) If $\mathsf{mgu}(\sigma, t) = \theta$
then $\Gamma \;/\; \sigma \doteq t : \kappa \dashv (\Gamma, \Delta)$
where Δ *has the form* $\alpha_1 = t_1, \ldots, \alpha_n = t_n$
and for all u *such that* $\Gamma \vdash u : \kappa$*, it is the case that* $[\Gamma, \Delta]u = \theta([\Gamma]u)$.

(2) If $\mathsf{mgu}(\sigma, t) = \bot$ *(that is, no most general unifier exists) then* $\Gamma \;/\; \sigma \doteq t : \kappa \dashv \bot$.

Proof. By induction on the structure of $[\Gamma]\sigma$ and $[\Gamma]t$.

- Case $[\Omega]\sigma = t = \mathsf{zero}$:

	$\mathsf{mgu}(\mathsf{zero}, \mathsf{zero}) = \cdot$	By properties of unification
	$\Gamma \;/\; \mathsf{zero} \doteq \mathsf{zero} : \mathbb{N} \dashv \Gamma$	By rule ElimeqZero
☞	$\Gamma \;/\; \mathsf{zero} \doteq \mathsf{zero} : \mathbb{N} \dashv \Gamma, \Delta$	where $\Delta = \cdot$
☞	Suppose $\Gamma \vdash u : \kappa'$.	
	$[\Gamma, \Delta]u = [\Gamma]u$	where $\Delta = \cdot$
	$= \theta([\Gamma]u)$	where θ is the identity

- Case $\sigma = \mathsf{succ}(\sigma')$ and $t = \mathsf{succ}(t')$:
  - Case $\mathsf{mgu}(\mathsf{succ}(\sigma'), \mathsf{succ}(t')) = \theta$:

	$\mathsf{mgu}(\sigma', t') = \mathsf{mgu}(\mathsf{succ}(\sigma'), \mathsf{succ}(t')) = \theta$	By properties of unification
	$\mathsf{succ}(\sigma') = [\Gamma]\mathsf{succ}(\sigma')$	Given
	$= \mathsf{succ}([\Gamma]\sigma')$	By definition of substitution
	$\sigma' = [\Gamma]\sigma'$	By injectivity of successor
	$\mathsf{succ}(t') = [\Gamma]\mathsf{succ}(t')$	Given
	$= \mathsf{succ}([\Gamma]t')$	By definition of substitution
	$t' = [\Gamma]t'$	By injectivity of successor
	$\Gamma \;/\; \sigma' \doteq t' : \mathbb{N} \dashv \Gamma, \Delta$	By i.h.
☞	$[\Gamma, \Delta]u = \theta([\Gamma]u)$ for all u such that …	″
☞	$\Gamma \;/\; \mathsf{succ}(\sigma') \doteq \mathsf{succ}(t') : \mathbb{N} \dashv \Gamma, \Delta$	By rule ElimeqSucc

  - Case $\mathsf{mgu}(\mathsf{succ}(\sigma'), \mathsf{succ}(t')) = \bot$:

	$\mathsf{mgu}(\sigma', t') = \mathsf{mgu}(\mathsf{succ}(\sigma'), \mathsf{succ}(t')) = \bot$	By properties of unification
	$\mathsf{succ}(\sigma') = [\Gamma]\mathsf{succ}(\sigma')$	Given
	$= \mathsf{succ}([\Gamma]\sigma')$	By definition of substitution
	$\sigma' = [\Gamma]\sigma'$	By injectivity of successor
	$\mathsf{succ}(t') = [\Gamma]\mathsf{succ}(t')$	Given
	$= \mathsf{succ}([\Gamma]t')$	By definition of substitution
	$t' = [\Gamma]t'$	By injectivity of successor
	$\Gamma \;/\; \sigma' \doteq t' : \mathbb{N} \dashv \bot$	By i.h.
☞	$\Gamma \;/\; \mathsf{succ}(\sigma') \doteq \mathsf{succ}(t') : \mathbb{N} \dashv \bot$	By rule ElimeqSucc

- Case $\sigma = \sigma_1 \oplus \sigma_2$ and $t = t_1 \oplus t_2$:
  First we establish some properties of the subterms:

$\sigma_1 \oplus \sigma_2 = [\Gamma](\sigma_1 \oplus \sigma_2)$ Given
$= [\Gamma]\sigma_1 \oplus [\Gamma]\sigma_2$ By definition of substitution
☞ $[\Gamma]\sigma_1 = \sigma_1$ By injectivity of $\oplus$
☞ $[\Gamma]\sigma_2 = \sigma_2$ By injectivity of $\oplus$
$t_1 \oplus t_2 = [\Gamma](t_1 \oplus t_2)$ Given
$= [\Gamma]t_1 \oplus [\Gamma]t_2$ By definition of substitution
☞ $[\Gamma]t_1 = t_1$ By injectivity of $\oplus$
☞ $[\Gamma]t_2 = t_2$ By injectivity of $\oplus$

– Subcase $\mathsf{mgu}(\sigma, t) = \bot$:

∗ Subcase $\mathsf{mgu}(\sigma_1, t_1) = \bot$:

$\Gamma \ / \ \sigma_1 \stackrel{?}{=} t_1 : \kappa \dashv \bot$ By i.h.
$\Gamma \ / \ \sigma_1 \oplus \sigma_2 \stackrel{?}{=} t_1 \oplus t_2 : \kappa \dashv \bot$ By rule ElimeqBinBot

∗ Subcase $\mathsf{mgu}(\sigma_1, t_1) = \theta_1$ and $\mathsf{mgu}(\theta_1(\sigma_2), \theta_1(t_2)) = \bot$:

$\Gamma \ / \ \sigma_1 \stackrel{?}{=} t_1 : \kappa \dashv \Gamma, \Delta_1$ By i.h.
$[\Gamma, \Delta_1]u = \theta_1([\Gamma]u)$ for all u such that ... ″
$[\Gamma, \Delta_1]\sigma_2 = \theta_1([\Gamma]\sigma_2)$ Above line with σ_2 as u
$= \theta_1(\sigma_2)$ $[\Gamma]\sigma_2 = \sigma_2$
$[\Gamma, \Delta_1]t_2 = \theta_1([\Gamma]t_2)$ Above line with t_2 as u
$= \theta_1(t_2)$ Since $[\Gamma]\sigma_2 = \sigma_2$
$\mathsf{mgu}([\Gamma, \Delta_1]\sigma_2, [\Gamma, \Delta_1]t_2) = \theta_2$ By transitivity of equality
$[\Gamma, \Delta_1][\Gamma, \Delta_1]\sigma_2 = [\Gamma, \Delta_1]\sigma_2$ By Lemma 29 (Substitution Monotonicity)
$[\Gamma, \Delta_1][\Gamma, \Delta_1]t_2 = [\Gamma, \Delta_1]t_2$ By Lemma 29 (Substitution Monotonicity)
$\Gamma, \Delta_1 \ / \ [\Gamma, \Delta_1]\sigma_2 \stackrel{?}{=} [\Gamma, \Delta_1]t_2 : \kappa \dashv \bot$ By i.h.
☞ $\Gamma \ / \ \sigma_1 \oplus \sigma_2 \stackrel{?}{=} t_1 \oplus t_2 : \kappa \dashv \bot$ By rule ElimeqBin

– Subcase $\mathsf{mgu}(\sigma, t) = \theta$:

$\mathsf{mgu}(\sigma_1 \oplus \sigma_2, t_1 \oplus t_2) = \theta = \theta_2 \circ \theta_1$ By properties of unifiers
$\mathsf{mgu}(\sigma_1, t_1) = \theta_1$ ″
$\mathsf{mgu}(\theta_1(\sigma_2), \theta_1(t_2)) = \theta_2$ ″
$\Gamma \ / \ \sigma_1 \stackrel{?}{=} t_1 : \kappa \dashv \Gamma, \Delta_1$ By i.h.
∗ $[\Gamma, \Delta_1]u = \theta_1([\Gamma]u)$ for all u such that ... ″
$[\Gamma, \Delta_1]\sigma_2 = \theta_1([\Gamma]\sigma_2)$ Above line with σ_2 as u
$= \theta_1(\sigma_2)$ $[\Gamma]\sigma_2 = \sigma_2$
$[\Gamma, \Delta_1]t_2 = \theta_1([\Gamma]t_2)$ Above line with t_2 as u
$= \theta_1(t_2)$ $[\Gamma]\sigma_2 = \sigma_2$
$\mathsf{mgu}([\Gamma, \Delta_1]\sigma_2, [\Gamma, \Delta_1]t_2) = \theta_2$ By transitivity of equality
$[\Gamma, \Delta_1][\Gamma, \Delta_1]\sigma_2 = [\Gamma, \Delta_1]\sigma_2$ By Lemma 29 (Substitution Monotonicity)
$[\Gamma, \Delta_1][\Gamma, \Delta_1]t_2 = [\Gamma, \Delta_1]t_2$ By Lemma 29 (Substitution Monotonicity)
$\Gamma, \Delta_1 \ / \ [\Gamma, \Delta_1]\sigma_2 \stackrel{?}{=} [\Gamma, \Delta_1]t_2 : \kappa \dashv \Gamma, \Delta_1, \Delta_2$ By i.h.
∗∗ $[\Gamma, \Delta_1, \Delta_2]u' = \theta_2([\Gamma, \Delta_1]u')$ for all u' such that ... ″
☞ $\Gamma \ / \ \sigma_1 \oplus \sigma_2 \stackrel{?}{=} t_1 \oplus t_2 : \kappa \dashv \Gamma, \Delta_1, \Delta_2$ By rule ElimeqBin

☞ Suppose $\Gamma \vdash u : \kappa'$.
$[\Gamma, \Delta_1, \Delta_2]u = \theta_2([\Gamma, \Delta_1]u)$ By ∗∗
$= \theta_2(\theta_1([\Gamma]u))$ By ∗
$= \theta([\Gamma]u)$ $\theta = \theta_2 \circ \theta_1$

- Case $\sigma = \alpha$:

– Subcase $\alpha \in \mathrm{FV}(t)$:

$\mathsf{mgu}(\alpha, t) = \bot$ By properties of unification
☞ $\Gamma \ / \ \alpha \stackrel{?}{=} t : \kappa \dashv \bot$ By rule ElimeqUvarL⊥

– Subcase $\alpha \notin \mathrm{FV}(t)$:

$\mathsf{mgu}(\alpha, t) = [t/\alpha]$ By properties of unification
$(\alpha = t') \notin \Gamma$ $[\Gamma]\alpha = \alpha$
☞ $\Gamma \;/\; \alpha \doteq t : \kappa \dashv \Gamma, \alpha = t$ By rule ElimeqUvarL
☞ Suppose $\Gamma \vdash u : \kappa'$.

$$\begin{aligned}[\Gamma, \alpha = t]u &= [\Gamma]([t/\alpha]u) && \text{By definition of substitution}\\ &= \big[[\Gamma]t/\alpha\big][\Gamma]u && \text{By properties of substitution}\\ &= [t/\alpha][\Gamma]u && [\Gamma]t = t\end{aligned}$$

- Case $t = \alpha$: Similar to previous case. □

Lemma 93 (Substitution Upgrade)**.**
If Δ *has the form* $\alpha_1 = t_1, \ldots, \alpha_n = t_n$
and, for all u *such that* $\Gamma \vdash u : \kappa$*, it is the case that* $[\Gamma, \Delta]u = \theta([\Gamma]u)$*,*
then:

(i) If $\Gamma \vdash A$ *type then* $[\Gamma, \Delta]A = \theta([\Gamma]A)$.

(ii) If $\Gamma \longrightarrow \Omega$ *then* $[\Omega]\Gamma = \theta([\Omega]\Gamma)$.

(iii) If $\Gamma \longrightarrow \Omega$ *then* $[\Omega, \Delta](\Gamma, \Delta) = \theta([\Omega]\Gamma)$.

(iv) If $\Gamma \longrightarrow \Omega$ *then* $[\Omega, \Delta]e = \theta([\Omega]e)$.

Proof. Part (i): By induction on the given derivation, using the given "for all" at the leaves.
Part (ii): By induction on the given derivation, using part (i) in the $\longrightarrow$Var case.
Part (iii): By induction on Δ. In the base case ($\Delta = \cdot$), use part (ii). Otherwise, use the i.h. and the definition of context substitution.
Part (iv): By induction on e, using part (i) in the $e = (e_0 : A)$ case. □

Lemma 94 (Completeness of Propequiv)**.**
Given $\Gamma \longrightarrow \Omega$
and $\Gamma \vdash P$ *prop and* $\Gamma \vdash Q$ *prop*
and $[\Omega]P = [\Omega]Q$
then $\Gamma \vdash [\Gamma]P \equiv [\Gamma]Q \dashv \Delta$
where $\Delta \longrightarrow \Omega'$ *and* $\Omega \longrightarrow \Omega'$.

Proof. By induction on the given derivations. There is only one possible case:

- **Case** $\dfrac{\Gamma \vdash \sigma_1 : \mathbb{N} \qquad \Gamma \vdash \sigma_2 : \mathbb{N}}{\Gamma \vdash \sigma_1 = \sigma_2 \; prop}\,\mathsf{EqProp} \qquad \dfrac{\Gamma \vdash \tau_1 : \mathbb{N} \qquad \Gamma \vdash \tau_2 : \mathbb{N}}{\Gamma \vdash \tau_1 = \tau_2 \; prop}\,\mathsf{EqProp}$

	$[\Omega](\sigma_1 = \sigma_2) = [\Omega](\tau_1 = \tau_2)$	Given
	$[\Omega]\sigma_1 = [\Omega]\tau_1$	Definition of substitution
	$[\Omega]\sigma_2 = [\Omega]\tau_2$	$''$
	$\Gamma \vdash \sigma_1 : \mathbb{N}$	Subderivation
	$\Gamma \vdash \tau_1 : \mathbb{N}$	Subderivation
	$\Gamma \vdash [\Gamma]\sigma_1 \doteq [\Gamma]\sigma_2 : \mathbb{N} \dashv \Theta$	By Lemma 91 (Completeness of Checkeq)
	$\Theta \longrightarrow \Omega_0$	$''$
	$\Omega \longrightarrow \Omega_0$	$''$
	$\Gamma \vdash \sigma_2 : \mathbb{N}$	Subderivation
	$\Theta \vdash \sigma_2 : \mathbb{N}$	By Lemma 36 (Extension Weakening (Sorts))
	$\Theta \vdash \tau_2 : \mathbb{N}$	Similarly
	$\Theta \vdash [\Theta]\tau_1 \doteq [\Theta]\tau_2 : \mathbb{N} \dashv \Delta$	By Lemma 91 (Completeness of Checkeq)
☞	$\Delta \longrightarrow \Omega_0$	$''$
	$\Omega_0 \longrightarrow \Omega'$	$''$
	$[\Theta]\tau_1 = [\Theta][\Gamma]\tau_1$	By Lemma 29 (Substitution Monotonicity) (i)
	$[\Theta]\tau_2 = [\Theta][\Gamma]\tau_2$	$''$
	$\Theta \vdash [\Theta][\Gamma]\tau_1 \doteq [\Theta][\Gamma]\tau_2 : \mathbb{N} \dashv \Delta$	By above equalities
☞	$\Omega \longrightarrow \Omega'$	By Lemma 33 (Extension Transitivity)
	$\Gamma \vdash ([\Gamma]\sigma_1 = [\Theta]\sigma_2) \equiv ([\Gamma]\tau_1 = [\Theta]\tau_2) \dashv \Gamma$	By $\equiv$PropEq
☞	$\Gamma \vdash ([\Gamma]\sigma_1 = [\Gamma]\sigma_2) \equiv ([\Gamma]\tau_1 = [\Gamma]\tau_2) \dashv \Gamma$	By above equalities □

Lemma 95 (Completeness of Checkprop)**.**
If $\Gamma \longrightarrow \Omega$ *and* $\mathsf{dom}(\Gamma) = \mathsf{dom}(\Omega)$
and $\Gamma \vdash P$ *prop*
and $[\Gamma]P = P$
and $[\Omega]\Gamma \vdash [\Omega]P$ *true*
then $\Gamma \vdash P$ *true* $\dashv \Delta$
where $\Delta \longrightarrow \Omega'$ *and* $\Omega \longrightarrow \Omega'$ *and* $\mathsf{dom}(\Delta) = \mathsf{dom}(\Omega')$.

Proof. Only one rule, DeclCheckpropEq, can derive $[\Omega]\Gamma \vdash [\Omega]P$ *true*, so by inversion, P has the form $(t_1 = t_2)$ where $[\Omega]t_1 = [\Omega]t_2$.
By inversion on $\Gamma \vdash (t_1 = t_2)$ *prop*, we have $\Gamma \vdash t_1 : \mathbb{N}$ and $\Gamma \vdash t_2 : \mathbb{N}$.
Then by Lemma 91 (Completeness of Checkeq), $\Gamma \vdash [\Gamma]t_1 \doteq [\Gamma]t_2 : \mathbb{N} \dashv \Delta$ where $\Delta \longrightarrow \Omega'$ and $\Omega \longrightarrow \Omega'$.
By CheckpropEq, $\Gamma \vdash (t_1 = t_2)$ *true* $\dashv \Delta$. □

L′.2 Completeness of Equivalence and Subtyping

Lemma 96 (Completeness of Equiv)**.**
If $\Gamma \longrightarrow \Omega$ *and* $\Gamma \vdash A$ *type and* $\Gamma \vdash B$ *type*
and $[\Omega]A = [\Omega]B$
then there exist Δ *and* Ω' *such that* $\Delta \longrightarrow \Omega'$ *and* $\Omega \longrightarrow \Omega'$ *and* $\Gamma \vdash [\Gamma]A \equiv [\Gamma]B \dashv \Delta$.

Proof. By induction on the derivations of $\Gamma \vdash A$ *type* and $\Gamma \vdash B$ *type*.

We distinguish cases of the rule concluding the first derivation. In the first four cases (ImpliesWF, WithWF, ForallWF, ExistsWF), it follows from $[\Omega]A = [\Omega]B$ and the syntactic invariant that Ω substitutes terms t (rather than types A) that the second derivation is concluded by the *same* rule. Moreover, if none of these three rules concluded the first derivation, the rule concluding the second derivation must *not* be ImpliesWF, WithWF, ForallWF or ExistsWF either.

Because Ω is predicative, the head connective of $[\Gamma]A$ must be the same as the head connective of $[\Omega]A$.

We distinguish cases that are *impossible* (impossible), **fully written out**, and similar to fully-written-out cases. For the lower-right case, where both $[\Gamma]A$ and $[\Gamma]B$ have a binary connective $\oplus$, it must be the same connective.

The Vec type is omitted from the table, but can be treated similarly to $\supset$ and $\wedge$.

$[\Gamma]A$ \ $[\Gamma]B$	$\supset$	$\wedge$	$\forall\beta.\,B'$	$\exists\beta.\,B'$	1	α	$\hat{\beta}$	$B_1 \oplus B_2$
$\supset$	**Implies**	*imposs.*	*imposs.*	*imposs.*	*imposs.*	*imposs.*	*imposs.*	*imposs.*
$\wedge$	*imposs.*	With	*imposs.*	*imposs.*	*imposs.*	*imposs.*	*imposs.*	*imposs.*
$\forall\alpha.\,A'$	*imposs.*	*imposs.*	**Forall**	*imposs.*	*imposs.*	*imposs.*	*imposs.*	*imposs.*
$\exists\alpha.\,A'$	*imposs.*	*imposs.*	*imposs.*	Exists	*imposs.*	*imposs.*	*imposs.*	*imposs.*
1	*imposs.*	*imposs.*	*imposs.*	*imposs.*	**2.Units**	*imposs.*	2.BEx.Unit	*imposs.*
α	*imposs.*	*imposs.*	*imposs.*	*imposs.*	*imposs.*	**2.Uvars**	2.BEx.Uvar	*imposs.*
$\hat{\alpha}$	*imposs.*	*imposs.*	*imposs.*	*imposs.*	**2.AEx.Unit**	2.AEx.Uvar	**2.AEx.SameEx** **2.AEx.OtherEx**	**2.AEx.Bin**
$A_1 \oplus A_2$	*imposs.*	*imposs.*	*imposs.*	*imposs.*	*imposs.*	*imposs.*	2.BEx.Bin	2.Bins

- **Case** $\dfrac{\Gamma \vdash P\ \mathit{prop} \qquad \Gamma \vdash A_0\ \mathit{type}}{\Gamma \vdash P \supset A_0\ \mathit{type}}$ ImpliesWF

  This case of the rule concluding the first derivation coincides with the **Implies** entry in the table.

  We have $[\Omega]A = [\Omega]B$, that is, $[\Omega](P \supset A_0) = [\Omega]B$.
  Because Ω is predicative, B must have the form $Q \supset B_0$, where $[\Omega]P = [\Omega]Q$ and $[\Omega]A_0 = [\Omega]B_0$.

$\Gamma \vdash P\ \mathit{prop}$	Subderivation
$\Gamma \vdash A_0\ \mathit{type}$	Subderivation
$\Gamma \vdash Q \supset B_0\ \mathit{type}$	Given
$\Gamma \vdash Q\ \mathit{prop}$	By inversion on rule ImpliesWF
$\Gamma \vdash B_0\ \mathit{type}$	$''$
$\Gamma \vdash [\Gamma]P \equiv [\Gamma]Q \dashv \Theta$	By Lemma 94 (Completeness of Propequiv)
$\Theta \longrightarrow \Omega_0$	$''$
$\Omega \longrightarrow \Omega_0$	$''$
$\Gamma \longrightarrow \Theta$	By Lemma 48 (Prop Equivalence Extension)
$\Gamma \vdash A_0\ \mathit{type}$	Above
$\Gamma \vdash B_0\ \mathit{type}$	Above
$[\Omega]A_0 = [\Omega]B_0$	Above
$[\Omega_0]A_0 = [\Omega_0]B_0$	By Lemma 55 (Completing Completeness) (ii) twice
$\Gamma \vdash [\Gamma]A_0 \equiv [\Gamma]B_0 \dashv \Delta$	By i.h.
☞ $\Delta \longrightarrow \Omega'$	$''$
$\Omega_0 \longrightarrow \Omega'$	$''$
☞ $\Omega \longrightarrow \Omega'$	By Lemma 33 (Extension Transitivity)
$\Gamma \vdash ([\Gamma]P \supset [\Gamma]A_0) \equiv ([\Gamma]Q \supset [\Gamma]B_0) \dashv \Delta$	By $\equiv\supset$
☞ $\Gamma \vdash [\Gamma](P \supset A_0) \equiv [\Gamma](Q \supset B_0) \dashv \Delta$	By definition of substitution

- **Case** WithWF: Similar to the ImpliesWF case, coinciding with the With entry in the table.

- **Case** $\dfrac{\Gamma, \alpha : \kappa \vdash A_0\ \mathit{type}}{\Gamma \vdash \forall\alpha : \kappa.\, A_0\ \mathit{type}}$ ForallWF

  This case coincides with the **Forall** entry in the table.

	$\Gamma \longrightarrow \Omega$	Given
	$\Gamma, \alpha : \kappa \longrightarrow \Omega, \alpha : \kappa$	By $\longrightarrow$Uvar
	$\Gamma, \alpha : \kappa \vdash A_0$ *type*	Subderivation
	$B = \forall \alpha : \kappa.\ B_0$	Ω predicative
	$[\Omega]A_0 = [\Omega]B_0$	From definition of substitution
	$\Gamma, \alpha : \kappa \vdash [\Gamma]A_0 \equiv [\Gamma]B_0 \dashv \Delta_0$	By i.h.
	$\Delta_0 \longrightarrow \Omega_0$	$''$
	$\Omega, \alpha : \kappa \longrightarrow \Omega_0$	$''$
☞	$\Omega \longrightarrow \Omega'$ and $\Omega_0 = (\Omega', \alpha : \kappa, \dots)$	By Lemma 22 (Extension Inversion) (i)
	$\Delta_0 = (\Delta, \alpha : \kappa, \Delta')$	By Lemma 22 (Extension Inversion) (i)
☞	$\Delta \longrightarrow \Omega'$	$''$
	$\Gamma \vdash \forall \alpha : \kappa.\ [\Gamma]A_0 \equiv \forall \alpha : \kappa.\ [\Gamma]B_0 \dashv \Delta$	By $\equiv\forall$
☞	$\Gamma \vdash [\Gamma](\forall \alpha : \kappa.\ A_0) \equiv [\Gamma](\forall \alpha : \kappa.\ B_0) \dashv \Delta$	By definition of substitution

- **Case** ExistsWF: Similar to the ForallWF case. (This is the Exists entry in the table.)
- **Case** BinWF: If BinWF also concluded the second derivation, then the proof is similar to the ImpliesWF case, but on the first premise, using the i.h. instead of Lemma 94 (Completeness of Propequiv). This is the 2.Bins entry in the lower right corner of the table.

  If BinWF did not conclude the second derivation, we are in the **2.AEx.Bin** or 2.BEx.Bin entries; see below.

In the remainder, we cover the 4×4 region in the lower right corner, starting from **2.Units**. We already handled the 2.Bins entry in the extreme lower right corner. At this point, we split on the forms of $[\Gamma]A$ and $[\Gamma]B$ instead; in the remaining cases, one or both types is atomic (e.g. **2.Uvars**, **2.AEx.Bin**) and we will not need to use the induction hypothesis.

- **Case 2.Units:** $[\Gamma]A = [\Gamma]B = 1$

☞	$\Gamma \vdash 1 \equiv 1 \dashv \Gamma$	By $\equiv$Unit
	$\Gamma \longrightarrow \Omega$	Given
	Let $\Omega' = \Omega'$.	
☞	$\Delta \longrightarrow \Omega$	$\Delta = \Gamma$
☞	$\Omega \longrightarrow \Omega'$	By Lemma 32 (Extension Reflexivity) and $\Omega' = \Omega$

- **Case 2.Uvars:** $[\Gamma]A = [\Gamma]B = \alpha$

	$\Gamma \longrightarrow \Omega$	Given
	Let $\Omega' = \Omega'$.	
☞	$\Gamma \vdash \alpha \equiv \alpha \dashv \Gamma$	By $\equiv$Var
☞	$\Delta \longrightarrow \Omega$	$\Delta = \Gamma$
☞	$\Omega \longrightarrow \Omega'$	By Lemma 32 (Extension Reflexivity) and $\Omega' = \Omega$

- **Case 2.AExUnit:** $[\Gamma]A = \hat{\alpha}$ and $[\Gamma]B = 1$

	$\Gamma \longrightarrow \Omega$	Given
	$1 = [\Omega]1$	By definition of substitution
	$\hat{\alpha} \notin \mathrm{FV}(1)$	By definition of FV$(-)$
	$[\Omega]\Gamma \vdash [\Omega]\hat{\alpha} \leq^{\pm} [\Omega]1$	Given
	$\Gamma \vdash \hat{\alpha} := 1 : \star \dashv \Delta$	By Lemma 90 (Completeness of Instantiation) (1)
☞	$\Omega \longrightarrow \Omega'$	$''$
☞	$\Delta \longrightarrow \Omega'$	$''$
	$1 = [\Gamma]1$	By definition of substitution
	$\hat{\alpha} \notin \mathrm{FV}(1)$	By definition of FV$(-)$
☞	$\Gamma \vdash \hat{\alpha} \equiv 1 \dashv \Delta$	By $\equiv$InstantiateL

- **Case 2.BExUnit:** $[\Gamma]A = 1$ and $[\Gamma]B = \hat{\alpha}$

  Symmetric to the **2.AExUnit** case.

- **Case 2.AEx.Uvar:** $[\Gamma]A = \hat{\alpha}$ and $[\Gamma]B = \alpha$

  Similar to the **2.AEx.Unit** case, using $\beta = [\Omega]\beta = [\Gamma]\beta$ and $\hat{\alpha} \notin \mathrm{FV}(\beta)$.

- **Case 2.BExUvar:** $[\Gamma]A = 1$ and $[\Gamma]B = \hat{\alpha}$

  Symmetric to the **2.AExUvar** case.

- **Case 2.AEx.SameEx:** $[\Gamma]A = \hat{\alpha} = \hat{\beta} = [\Gamma]B$

	$\Gamma \vdash \hat{\alpha} \equiv \hat{\alpha} \dashv \Gamma$	By $\equiv$Exvar ($\hat{\alpha} = \hat{\beta}$)
	$[\Gamma]\hat{\alpha} = \hat{\alpha}$	$\hat{\alpha}$ unsolved in Γ
☞	$\Gamma \vdash [\Gamma]\hat{\alpha} \equiv [\Gamma]\hat{\beta} \dashv \Gamma$	By above equality + $\hat{\alpha} = \hat{\beta}$
	$\Gamma \longrightarrow \Omega$	Given
☞	$\Delta \longrightarrow \Omega$	$\Delta = \Gamma$
	Let $\Omega' = \Omega$.	
☞	$\Omega \longrightarrow \Omega'$	By Lemma 32 (Extension Reflexivity) and $\Omega' = \Omega$

- **Case 2.AEx.OtherEx:** $[\Gamma]A = \hat{\alpha}$ and $[\Gamma]B = \hat{\beta}$ and $\hat{\alpha} \neq \hat{\beta}$

  Either $\hat{\alpha} \in \mathrm{FV}([\Gamma]\hat{\beta})$, or $\hat{\alpha} \notin \mathrm{FV}([\Gamma]\hat{\beta})$.

  - $\hat{\alpha} \in \mathrm{FV}([\Gamma]\hat{\beta})$:
    We have $\hat{\alpha} \preceq [\Gamma]\hat{\beta}$.
    Therefore $\hat{\alpha} = [\Gamma]\hat{\beta}$, or $\hat{\alpha} \prec [\Gamma]\hat{\beta}$.
    But we are in Case 2.AEx.*Other*Ex, so the former is impossible.
    Therefore, $\hat{\alpha} \prec [\Gamma]\hat{\beta}$.
    By a property of substitutions, $[\Omega]\hat{\alpha} \prec [\Omega][\Gamma]\hat{\beta}$.
    Since $\Gamma \longrightarrow \Omega$, by Lemma 29 (Substitution Monotonicity) (iii), $[\Omega][\Gamma]\hat{\beta} = [\Omega]\hat{\beta}$, so $[\Omega]\hat{\alpha} \prec [\Omega]\hat{\beta}$.
    But it is given that $[\Omega]\hat{\alpha} = [\Omega]\hat{\beta}$, a contradiction.
  - $\hat{\alpha} \notin \mathrm{FV}([\Gamma]\hat{\beta})$:

	$\Gamma \vdash \hat{\alpha} := [\Gamma]\hat{\beta} : \star \dashv \Delta$	By Lemma 90 (Completeness of Instantiation)
☞	$\Gamma \vdash \hat{\alpha} \equiv [\Gamma]\hat{\beta} \dashv \Delta$	By $\equiv$InstantiateL
☞	$\Delta \longrightarrow \Omega'$	$''$
☞	$\Omega \longrightarrow \Omega'$	$''$

- **Case 2.AEx.Bin:** $[\Gamma]A = \hat{\alpha}$ and $[\Gamma]B = B_1 \oplus B_2$

  Since $[\Gamma]B$ is an arrow, it cannot be exactly $\hat{\alpha}$. By the same reasoning as in the previous case (2.AEx.OtherEx), $\hat{\alpha} \notin \mathrm{FV}([\Gamma]\hat{\beta})$.

	$\Gamma \vdash \hat{\alpha} := [\Gamma]B : \star \dashv \Delta$	By Lemma 90 (Completeness of Instantiation)
☞	$\Delta \longrightarrow \Omega'$	$''$
☞	$\Omega \longrightarrow \Omega'$	$''$
☞	$\Gamma \vdash \underbrace{[\Gamma]A}_{\hat{\alpha}} \equiv \underbrace{[\Gamma]B}_{B_1 \oplus B_2} \dashv \Delta$	By $\equiv$InstantiateL

- **Case 2.BEx.Bin:** $[\Gamma]A = A_1 \oplus A_2$ and $[\Gamma]B = \hat{\beta}$

  Symmetric to the **2.AEx.Bin** case, applying $\equiv$InstantiateR instead of $\equiv$InstantiateL. □

Theorem 9 (Completeness of Subtyping)**.**
If $\Gamma \longrightarrow \Omega$ *and* $\mathsf{dom}(\Gamma) = \mathsf{dom}(\Omega)$ *and* $\Gamma \vdash A$ *type and* $\Gamma \vdash B$ *type*
and $[\Omega]\Gamma \vdash [\Omega]A \leq^{\pm} [\Omega]B$
then there exist Δ *and* Ω' *such that* $\Delta \longrightarrow \Omega'$
and $\mathsf{dom}(\Delta) = \mathsf{dom}(\Omega')$
and $\Omega \longrightarrow \Omega'$
and $\Gamma \vdash [\Gamma]A <:^{\pm} [\Gamma]B \dashv \Delta$.

Proof. By induction on the number of $\forall/\exists$ quantifiers in $[\Omega]A$ and $[\Omega]B$.

It is straightforward to show $\mathsf{dom}(\Delta) = \mathsf{dom}(\Omega')$; for examples of the necessary reasoning, see the proof of Theorem 11 (Completeness of Algorithmic Typing).

We have $[\Omega]\Gamma \vdash [\Omega]A \leq^{\pm} [\Omega]B$.

- **Case** $\dfrac{[\Omega]\Gamma \vdash [\Omega]A \; \mathit{type} \qquad \mathit{nonpos}([\Omega]A)}{[\Omega]\Gamma \vdash [\Omega]A \leq^{-} \underbrace{[\Omega]A}_{[\Omega]B}} \leq\mathsf{Refl}-$

  First, we observe that, since applying Ω as a substitution leaves quantifiers alone, the quantifiers that head A must also head B. For convenience, we alpha-vary B to quantify over the same variables as A.

  - If A is headed by $\forall$, then $[\Omega]A = (\forall\alpha : \kappa.\, [\Omega]A_0) = (\forall\alpha : \kappa.\, [\Omega]B_0) = [\Omega]B$.
    Let $\Gamma_0 = (\Gamma, \alpha : \kappa, \blacktriangleright_{\hat\alpha}, \hat\alpha : \kappa)$.
    Let $\Omega_0 = (\Omega, \alpha : \kappa, \blacktriangleright_{\hat\alpha}, \hat\alpha : \kappa = \alpha)$.
    - If $\mathrm{pol}(A_0) \in \{-, 0\}$, then:
      (We elide the straightforward use of lemmas about context extension.)

	$[\Omega_0]\Gamma_0 \vdash [\Omega]A_0 \leq^{-} [\Omega]A_0$	By $\leq$Refl$-$
	$[\Omega_0]\Gamma_0 \vdash [\Omega_0][\hat\alpha/\alpha]A_0 \leq^{-} A_0$	By def. of subst.
	$\Delta_0 \longrightarrow \Omega_0'$	By i.h. (fewer quantifiers)
	$\Omega_0 \longrightarrow \Omega_0'$	$''$
	$\Gamma_0 \vdash [\Gamma_0][\hat\alpha/\alpha]A_0 <:^{-} [\Gamma]B_0 \dashv \Delta_0$	$''$
	$\Gamma_0 \vdash [\hat\alpha/\alpha][\Gamma_0]A_0 <:^{-} [\Gamma]B_0 \dashv \Delta_0$	$\hat\alpha$ unsolved in Γ_0
	$\Gamma_0 \vdash [\hat\alpha/\alpha][\Gamma]A_0 <:^{-} [\Gamma]B_0 \dashv \Delta_0$	Γ_0 substitutes as Γ
	$\Gamma, \alpha : \kappa \vdash \forall\alpha : \kappa.\, [\Gamma]A_0 <:^{-} [\Gamma]B_0 \dashv \Delta, \alpha : \kappa, \Theta$	By $<:\forall$L
	$\Gamma \vdash \forall\alpha : \kappa.\, [\Gamma]A_0 <:^{-} \forall\alpha : \kappa.\, [\Gamma]B_0 \dashv \Delta$	By $<:\forall$R
☞	$\Gamma \vdash [\Gamma](\forall\alpha : \kappa.\, A_0) <:^{-} [\Gamma](\forall\alpha : \kappa.\, B_0) \dashv \Delta$	By def. of subst.
☞	$\Delta \longrightarrow \Omega$	By lemma
☞	$\Omega \longrightarrow \Omega_0'$	By lemma

    - If $\mathrm{pol}(A_0) = +$, then proceed as above, but apply $\leq$Refl$+$ instead of $\leq$Refl$-$, and apply $<:{}^{+}_{-}$L after applying the i.h. (Rule $<:{}^{+}_{-}$R also works.)
  - If A is not headed by $\forall$:
    We have $\mathit{nonneg}([\Omega]A)$. Therefore $\mathit{nonneg}(A)$, and thus A is not headed by $\exists$. Since the same quantifiers must also head B, the conditions in rule $<:$Equiv are satisfied.

	$\Gamma \longrightarrow \Omega$	Given
	$\Gamma \vdash [\Gamma]A \equiv [\Gamma]B \dashv \Delta$	By Lemma 96 (Completeness of Equiv)
☞	$\Delta \longrightarrow \Omega'$	$''$
☞	$\Omega \longrightarrow \Omega'$	$''$
☞	$\Gamma \vdash [\Gamma]A <:^{-} [\Gamma]B \dashv \Delta$	By $<:$Equiv

- **Case** $\leq$Refl$+$: Symmetric to the $\leq$Refl$-$ case, using $<:{}^{-}_{+}$L (or $<:{}^{-}_{+}$R), and $<:\exists$R/$<:\exists$L instead of $<:\forall$L/$<:\forall$R.

- **Case** $\dfrac{[\Omega]\Gamma \vdash \tau : \kappa \qquad [\Omega]\Gamma \vdash [\tau/\alpha][\Omega]A_0 \leq^{-} [\Omega]B}{[\Omega]\Gamma \vdash \underbrace{\forall\alpha : \kappa.\, [\Omega]A_0}_{[\Omega]A} \leq^{-} [\Omega]B} \leq\forall$L

  We begin by considering whether or not $[\Omega]B$ is headed by a universal quantifier.

  - $[\Omega]B = (\forall\beta : \kappa'.\, B')$:

    $[\Omega]\Gamma, \beta : \kappa' \vdash [\Omega]A \leq^{-} B'$ By Lemma 5 (Subtyping Inversion)
    The remaining steps are similar to the $\leq\forall$R case.

– $[\Omega]B$ not headed by $\forall$:

$$\begin{array}{rl}
[\Omega]\Gamma \vdash \tau : \kappa & \text{Subderivation}\\
\Gamma \longrightarrow \Omega & \text{Given}\\
\Gamma, \blacktriangleright_{\hat{\alpha}} \longrightarrow \Omega, \blacktriangleright_{\hat{\alpha}} & \text{By} \longrightarrow\text{Marker}\\
\Gamma, \blacktriangleright_{\hat{\alpha}}, \hat{\alpha} : \kappa \longrightarrow \underbrace{\Omega, \blacktriangleright_{\hat{\alpha}}, \hat{\alpha} : \kappa = \tau}_{\Omega_0} & \text{By} \longrightarrow\text{Solve}\\
[\Omega]\Gamma = [\Omega_0](\Gamma, \blacktriangleright_{\hat{\alpha}}, \hat{\alpha} : \kappa) & \text{By definition of context application (lines 16, 13)}
\end{array}$$

$$\begin{array}{rl}
[\Omega]\Gamma \vdash [\tau/\alpha][\Omega]A_0 \leq^- [\Omega]B & \text{Subderivation}\\
[\Omega_0](\Gamma, \blacktriangleright_{\hat{\alpha}}, \hat{\alpha} : \kappa) \vdash [\tau/\alpha][\Omega]A_0 \leq^- [\Omega]B & \text{By above equality}\\
[\Omega_0](\Gamma, \blacktriangleright_{\hat{\alpha}}, \hat{\alpha} : \kappa) \vdash \big[[\Omega_0]\hat{\alpha}/\alpha\big][\Omega]A_0 \leq^- [\Omega]B & \text{By definition of substitution}\\
[\Omega_0](\Gamma, \blacktriangleright_{\hat{\alpha}}, \hat{\alpha} : \kappa) \vdash \big[[\Omega_0]\hat{\alpha}/\alpha\big][\Omega_0]A_0 \leq^- [\Omega_0]B & \text{By definition of substitution}\\
[\Omega_0](\Gamma, \blacktriangleright_{\hat{\alpha}}, \hat{\alpha} : \kappa) \vdash [\Omega_0][\hat{\alpha}/\alpha]A_0 \leq^- [\Omega_0]B & \text{By distributivity of substitution}
\end{array}$$

$$\begin{array}{rl}
\Gamma, \blacktriangleright_{\hat{\alpha}}, \hat{\alpha} \vdash [\Gamma, \blacktriangleright_{\hat{\alpha}}, \hat{\alpha} : \kappa][\hat{\alpha}/\alpha]A_0 <:^- [\Gamma, \blacktriangleright_{\hat{\alpha}}, \hat{\alpha} : \kappa]B \dashv \Delta_0 & \text{By i.h. } (A \text{ lost a quantifier})\\
\Delta_0 \longrightarrow \Omega'' & ''\\
\Omega_0 \longrightarrow \Omega'' & ''
\end{array}$$

$$\Gamma, \blacktriangleright_{\hat{\alpha}}, \hat{\alpha} : \kappa \vdash [\Gamma][\hat{\alpha}/\alpha]A_0 <:^- [\Gamma]B \dashv \Delta_0 \quad \text{By definition of substitution}$$

$$\begin{array}{rl}
\Gamma, \blacktriangleright_{\hat{\alpha}}, \hat{\alpha} : \kappa \longrightarrow \Delta_0 & \text{By Lemma 50 (Subtyping Extension)}\\
\Delta_0 = (\Delta, \blacktriangleright_{\hat{\alpha}}, \Theta) & \text{By Lemma 22 (Extension Inversion) (ii)}\\
\Gamma \longrightarrow \Delta & ''\\
\Omega'' = (\Omega', \blacktriangleright_{\hat{\alpha}}, \Omega_Z) & \text{By Lemma 22 (Extension Inversion) (ii)}\\
\Delta \longrightarrow \Omega' & ''\\
\Omega_0 \longrightarrow \Omega'' & \text{Above}\\
\Omega, \blacktriangleright_{\hat{\alpha}}, \hat{\alpha} : \kappa = \tau \longrightarrow \Omega', \blacktriangleright_{\hat{\alpha}}, \Omega_Z & \text{By above equalities}\\
\Omega \longrightarrow \Omega' & \text{By Lemma 22 (Extension Inversion) (ii)}
\end{array}$$

$$\begin{array}{rl}
\Gamma, \blacktriangleright_{\hat{\alpha}}, \hat{\alpha} : \kappa \vdash [\Gamma][\hat{\alpha}/\alpha]A_0 <:^- [\Gamma]B \dashv \Delta, \blacktriangleright_{\hat{\alpha}}, \Theta & \text{By above equality } \Delta_0 = (\Delta, \blacktriangleright_{\hat{\alpha}}, \Theta)\\
\Gamma, \blacktriangleright_{\hat{\alpha}}, \hat{\alpha} : \kappa \vdash [\hat{\alpha}/\alpha][\Gamma]A_0 <:^- [\Gamma]B \dashv \Delta, \blacktriangleright_{\hat{\alpha}}, \Theta & \text{By def. of subst. } ([\Gamma]\hat{\alpha} = \hat{\alpha} \text{ and } [\Gamma]\alpha = \alpha)\\
[\Gamma]B \text{ not headed by } \forall & \text{From the case assumption}\\
\Gamma \vdash \forall\alpha : \kappa.\, [\Gamma]A_0 <:^- [\Gamma]B \dashv \Delta & \text{By } <:\forall L\\
\Gamma \vdash [\Gamma](\forall\alpha : \kappa.\, A_0) <:^- [\Gamma]B \dashv \Delta & \text{By definition of substitution}
\end{array}$$

- **Case** $\dfrac{[\Omega]\Gamma, \beta : \kappa \vdash [\Omega]A \leq^- [\Omega]B_0}{[\Omega]\Gamma \vdash [\Omega]A \leq^- \underbrace{\forall\beta : \kappa.\, [\Omega]B_0}_{[\Omega]B}} \leq\forall R$

$$\begin{array}{rl}
B = \forall\beta : \kappa.\, B_0 & \Omega \text{ predicative}\\
[\Omega]\Gamma \vdash [\Omega]A \leq^- [\Omega]B & \text{Given}\\
[\Omega]\Gamma \vdash [\Omega]A \leq^- \forall\beta.\, [\Omega]B_0 & \text{By above equality}\\
[\Omega]\Gamma, \beta : \kappa \vdash [\Omega]A \leq^- [\Omega]B_0 & \text{Subderivation}\\
[\Omega, \beta : \kappa](\Gamma, \beta : \kappa) \vdash [\Omega, \beta : \kappa]A \leq^- [\Omega, \beta : \kappa]B_0 & \text{By definitions of substitution}\\
\Gamma, \beta : \kappa \vdash [\Gamma, \beta{:}\kappa]A <:^- [\Gamma, \beta{:}\kappa]B_0 \dashv \Delta' & \text{By i.h. } (B \text{ lost a quantifier})\\
\Delta' \longrightarrow \Omega_0' & ''\\
\Omega, \beta : \kappa \longrightarrow \Omega_0' & ''\\
\Gamma, \beta : \kappa \vdash [\Gamma]A <:^- [\Gamma]B_0 \dashv \Delta' & \text{By definition of substitution}\\
\Gamma, \beta : \kappa \longrightarrow \Delta' & \text{By Lemma 43 (Instantiation Extension)}\\
\Delta' = (\Delta, \beta : \kappa, \Theta) & \text{By Lemma 22 (Extension Inversion) (i)}\\
\Gamma \longrightarrow \Delta & ''\\
\Delta, \beta : \kappa, \Theta \longrightarrow \Omega_0' & \text{By } \Delta' \longrightarrow \Omega_0' \text{ and above equality}\\
\Omega_0' = (\Omega', \beta : \kappa, \Omega_R) & \text{By Lemma 22 (Extension Inversion) (i)}\\
\Delta \longrightarrow \Omega' & ''
\end{array}$$

$\Gamma, \beta : \kappa \vdash [\Gamma]A <:^{-} [\Gamma]B_0 \dashv \Delta, \beta : \kappa, \Theta$ By above equality
$\Omega, \beta : \kappa \longrightarrow \Omega', \beta : \kappa, \Omega_R$ By above equality
☞ $\Omega \longrightarrow \Omega'$ By Lemma 33 (Extension Transitivity)

$\Gamma \vdash [\Gamma]A <:^{-} \forall \beta : \kappa. [\Gamma]B_0 \dashv \Delta$ By $<:\forall R$
☞ $\Gamma \vdash [\Gamma]A <:^{-} [\Gamma](\forall \beta : \kappa. B_0) \dashv \Delta$ By definition of substitution

- **Case** $\dfrac{[\Omega]\Gamma, \alpha : \kappa \vdash [\Omega]A_0 \leq^{+} [\Omega]B}{[\Omega]\Gamma \vdash \underbrace{\exists \alpha : \kappa. [\Omega]A_0}_{[\Omega]A} \leq^{+} [\Omega]B} \leq\exists L$

$A = \exists \alpha : \kappa. A_0$ Ω predicative
$[\Omega]\Gamma \vdash [\Omega]A) \leq^{+} [\Omega]B$ Given
$[\Omega]\Gamma \vdash [\Omega]\exists \alpha : \kappa. A_0 \leq^{+} [\Omega]B$ By above equality
$[\Omega]\Gamma, \alpha : \kappa \vdash [\Omega]A_0 \leq^{+} [\Omega]B$ Subderivation
$[\Omega, \alpha : \kappa](\Gamma, \alpha : \kappa) \vdash [\Omega, \alpha : \kappa]A_0 \leq^{+} [\Omega, \alpha : \kappa]B$ By definitions of substitution
$\Gamma, \alpha : \kappa \vdash [\Gamma, \beta{:}\kappa]A_0 <:^{+} [\Gamma, \beta{:}\kappa]B \dashv \Delta'$ By i.h. (A lost a quantifier)
$\Delta' \longrightarrow \Omega_0'$ ″
$\Omega, \alpha : \kappa \longrightarrow \Omega_0'$ ″
$\Gamma, \alpha : \kappa \vdash [\Gamma]A <:^{+} [\Gamma]B_0 \dashv \Delta'$ By definition of substitution
$\Gamma, \alpha : \kappa \longrightarrow \Delta'$ By Lemma 43 (Instantiation Extension)
$\Delta' = (\Delta, \alpha : \kappa, \Theta)$ By Lemma 22 (Extension Inversion) (i)
$\Gamma \longrightarrow \Delta$ ″
$\Delta, \alpha : \kappa, \Theta \longrightarrow \Omega_0'$ By $\Delta' \longrightarrow \Omega_0'$ and above equality
$\Omega_0' = (\Omega', \alpha : \kappa, \Omega_R)$ By Lemma 22 (Extension Inversion) (i)
☞ $\Delta \longrightarrow \Omega'$ ″

$\Gamma, \alpha : \kappa \vdash [\Gamma]A_0 <:^{+} [\Gamma]B \dashv \Delta, \alpha : \kappa, \Theta$ By above equality
$\Omega, \alpha : \kappa \longrightarrow \Omega', \alpha : \kappa, \Omega_R$ By above equality
☞ $\Omega \longrightarrow \Omega'$ By Lemma 33 (Extension Transitivity)

$\Gamma \vdash \exists \alpha : \kappa. [\Gamma]A_0 <:^{+} [\Gamma]B \dashv \Delta$ By $<:\forall R$
☞ $\Gamma \vdash [\Gamma](\exists \alpha : \kappa. A_0) <:^{+} [\Gamma]B \dashv \Delta$ By definition of substitution

- **Case** $\dfrac{\Psi \vdash \tau : \kappa \qquad \Psi \vdash [\Omega]A \leq^{+} [\tau/\beta]B_0}{\Psi \vdash [\Omega]A \leq^{+} \underbrace{\exists \beta : \kappa. B_0}_{[\Omega]B}} \leq\exists R$

We consider whether $[\Omega]A$ is headed by an existential.

If $[\Omega]A = \exists \alpha : \kappa'. A'$:

$[\Omega]\Gamma, \alpha : \kappa' \vdash A' \leq^{+} [\Omega]B$ By Lemma 5 (Subtyping Inversion)

The remaining steps are similar to the $\leq\exists L$ case.

If $[\Omega]A$ not headed by $\exists$:

$[\Omega]\Gamma \vdash \tau : \kappa$ Subderivation
$\Gamma \longrightarrow \Omega$ Given
$\Gamma, \blacktriangleright_{\hat{\alpha}} \longrightarrow \Omega, \blacktriangleright_{\hat{\alpha}}$ By $\longrightarrow$Marker
$\Gamma, \blacktriangleright_{\hat{\alpha}}, \hat{\alpha} : \kappa \longrightarrow \underbrace{\Omega, \blacktriangleright_{\hat{\alpha}}, \hat{\alpha} : \kappa = \tau}_{\Omega_0}$ By $\longrightarrow$Solve
$[\Omega]\Gamma = [\Omega_0](\Gamma, \blacktriangleright_{\hat{\alpha}}, \hat{\alpha} : \kappa)$ By definition of context application (lines 16, 13)
$[\Omega]\Gamma \vdash [\Omega]A \leq^{+} [\tau/\beta][\Omega]B_0$ Subderivation
$[\Omega_0](\Gamma, \blacktriangleright_{\hat{\alpha}}, \hat{\alpha} : \kappa) \vdash [\Omega]A \leq^{+} [\tau/\beta][\Omega]B_0$ By above equality
$[\Omega_0](\Gamma, \blacktriangleright_{\hat{\alpha}}, \hat{\alpha} : \kappa) \vdash [\Omega]A \leq^{+} \left[[\Omega_0]\hat{\alpha}/\beta\right][\Omega]B_0$ By definition of substitution
$[\Omega_0](\Gamma, \blacktriangleright_{\hat{\alpha}}, \hat{\alpha} : \kappa) \vdash [\Omega_0]A \leq^{+} \left[[\Omega_0]\hat{\alpha}/\beta\right][\Omega_0]B_0$ By definition of substitution
$[\Omega_0](\Gamma, \blacktriangleright_{\hat{\alpha}}, \hat{\alpha} : \kappa) \vdash [\Omega_0]A \leq^{+} [\Omega_0][\hat{\alpha}/\beta]B_0$ By distributivity of substitution

$\Gamma, \blacktriangleright_{\hat\alpha}, \hat\alpha \vdash [\Gamma, \blacktriangleright_{\hat\alpha}, \hat\alpha : \kappa]A <:^+ [\Gamma, \blacktriangleright_{\hat\alpha}, \hat\alpha : \kappa][\hat\alpha/\beta]B_0 \dashv \Delta_0$ By i.h. (B lost a quantifier)
$\Delta_0 \longrightarrow \Omega''$ ″
$\Omega_0 \longrightarrow \Omega''$ ″

$\Gamma, \blacktriangleright_{\hat\alpha}, \hat\alpha : \kappa \vdash [\Gamma][\hat\alpha/\beta]B_0 <:^+ [\Gamma]B \dashv \Delta_0$ By definition of substitution
$\Gamma, \blacktriangleright_{\hat\alpha}, \hat\alpha : \kappa \longrightarrow \Delta_0$ By Lemma 50 (Subtyping Extension)
$\Delta_0 = (\Delta, \blacktriangleright_{\hat\alpha}, \Theta)$ By Lemma 22 (Extension Inversion) (ii)
$\Gamma \longrightarrow \Delta$ ″
$\Omega'' = (\Omega', \blacktriangleright_{\hat\alpha}, \Omega_Z)$ By Lemma 22 (Extension Inversion) (ii)
☞ $\Delta \longrightarrow \Omega'$ ″
$\Omega_0 \longrightarrow \Omega''$ Above
$\Omega, \blacktriangleright_{\hat\alpha}, \hat\alpha : \kappa = \tau \longrightarrow \Omega', \blacktriangleright_{\hat\alpha}, \Omega_Z$ By above equalities
☞ $\Omega \longrightarrow \Omega'$ By Lemma 22 (Extension Inversion) (ii)

$\Gamma, \blacktriangleright_{\hat\alpha}, \hat\alpha : \kappa \vdash [\Gamma]A <:^+ [\Gamma][\hat\alpha/\beta]B_0 \dashv \Delta, \blacktriangleright_{\hat\alpha}, \Theta$ By above equality $\Delta_0 = (\Delta, \blacktriangleright_{\hat\alpha}, \Theta)$
$\Gamma, \blacktriangleright_{\hat\alpha}, \hat\alpha : \kappa \vdash [\Gamma]A <:^+ [\hat\alpha/\beta][\Gamma]B_0 \dashv \Delta, \blacktriangleright_{\hat\alpha}, \Theta$ By def. of subst. ($[\Gamma]\hat\alpha = \hat\alpha$ and $[\Gamma]\beta = \beta$)
$[\Gamma]A$ not headed by $\exists$ From the case hypothesis
$\Gamma \vdash [\Gamma]A <:^+ \exists\beta : \kappa.\ [\Gamma]B_0 \dashv \Delta$ By $<:\exists R$
☞ $\Gamma \vdash [\Gamma]A <:^+ [\Gamma](\exists\beta : \kappa.\ B_0) \dashv \Delta$ By definition of substitution □

L′.3 Completeness of Typing

Theorem 10 (Completeness of Match Coverage)**.**

1. *If* $[\Omega]\Gamma \vdash [\Omega]\Pi$ *covers* $[\Omega]\vec{A}$ *and* $\Gamma \longrightarrow \Omega$ *and* $\Gamma \vdash \vec{A}$ **!** *types and* $[\Gamma]\vec{A} = \vec{A}$ *then* $\Gamma \vdash \Pi$ *covers* $\vec{A}$*.*
2. *If* $[\Omega]\Gamma \,/\, [\Omega]P \vdash [\Omega]\Pi$ *covers* $[\Omega]\vec{A}$ *and* $\Gamma \longrightarrow \Omega$ *and* $\Gamma \vdash \vec{A}$ **!** *types and* $[\Gamma]\vec{A} = \vec{A}$ *and* $[\Gamma]P = P$ *then* $\Gamma \,/\, P \vdash \Pi$ *covers* $\vec{A}$*.*

Proof. By mutual induction on the derivation of the given coverage rule.

1. • **Case**
$$\frac{}{[\Omega]\Gamma \vdash \cdot \Rightarrow e_1 \mid \ldots \textit{ covers } \cdot}\ \textsf{DeclCoversEmpty}$$
Apply CoversEmpty.
• **Cases** DeclCoversVar, DeclCovers1, DeclCovers×, DeclCovers+, DeclCovers∃, DeclCovers∧, DeclCoversVec:
Use the i.h. and apply the corresponding algorithmic coverage rule.

2. • **Case**
$$\frac{\theta = \mathsf{mgu}(t_1, t_2) \qquad [\theta][\Omega]\Gamma \vdash [\theta][\Omega]\Pi \textit{ covers } [\theta]\vec{A}}{[\Omega]\Gamma \,/\, [\Omega](t_1 = t_2) \vdash [\Omega]\Pi \textit{ covers } \vec{A}}\ \textsf{DeclCoversEq}$$

$\mathsf{mgu}(t_1, t_2) = \theta$ Premise
$\Gamma \;/\; t_1 \doteq t_2 : \kappa \dashv \Gamma, \Theta$ By Lemma 92 (Completeness of Elimeq) (1)
$\Gamma \;/\; [\Gamma]t_1 \doteq [\Gamma]t_2 : \kappa \dashv \Gamma, \Theta$ Follows from given assumption
$[\theta][\Omega]\Gamma \vdash [\theta][\Omega]\Pi \textit{ covers } [\theta]A_0, [\theta]\vec{A}$ Subderivation
$[\theta][\Omega]\Gamma = [\Omega, \Theta](\Gamma, \Theta)$ By Lemma 93 (Substitution Upgrade) (iii)
$[\theta][\Omega]\Pi = [\Omega, \Theta]\Pi$ By Lemma 93 (Substitution Upgrade) (iv)
$([\theta]A_0, [\theta]\vec{A}) = ([\Gamma, \Theta]\vec{A})$ By Lemma 93 (Substitution Upgrade) (i)
$[\Omega, \Theta](\Gamma, \Theta) \vdash [\Omega, \Theta]\Pi \textit{ covers } [\Gamma, \Theta]\vec{A}$ By above equalities
$\Gamma, \Theta \vdash [\Gamma, \Theta]\Pi \textit{ covers } [\Gamma, \Theta]\vec{A}$ By i.h.
☞ $\Gamma \,/\, t_1 = t_2 \vdash \Pi \textit{ covers } \vec{A}$ By CoversEq

- **Case** $\dfrac{\mathsf{mgu}(t_1, t_2) = \bot}{[\Omega]\Gamma \;/\; [\Omega](t_1 = t_2) \vdash [\Omega]\Pi \textit{ covers } \vec{A}}\,\mathsf{DeclCoversEqBot}$

	$\mathsf{mgu}(t_1, t_2) = \bot$	Premise
	$\Gamma \;/\; t_1 \doteq t_2 : \kappa \dashv \bot$	By Lemma 92 (Completeness of Elimeq) (2)
	$\Gamma \;/\; [\Gamma]t_1 \doteq [\Gamma]t_2 : \kappa \dashv \bot$	Follows from given assumption
☞	$\Gamma \;/\; t_1 = t_2 \vdash \Pi \textit{ covers } \vec{A}$	By CoversEqBot

□

Theorem 11 (Completeness of Algorithmic Typing)**.** *Given* $\Gamma \longrightarrow \Omega$ *such that* $\mathsf{dom}(\Gamma) = \mathsf{dom}(\Omega)$*:*

(i) *If* $\Gamma \vdash A\ p$ *type and* $[\Omega]\Gamma \vdash [\Omega]e \Leftarrow [\Omega]A\ p$ *and* $p' \sqsubseteq p$
then there exist Δ *and* Ω'
such that $\Delta \longrightarrow \Omega'$ *and* $\mathsf{dom}(\Delta) = \mathsf{dom}(\Omega')$ *and* $\Omega \longrightarrow \Omega'$
and $\Gamma \vdash e \Leftarrow [\Gamma]A\ p' \dashv \Delta$*.*

(ii) *If* $\Gamma \vdash A\ p$ *type and* $[\Omega]\Gamma \vdash [\Omega]e \Rightarrow A\ p$
then there exist Δ*,* Ω'*,* A'*, and* $p' \sqsubseteq p$
such that $\Delta \longrightarrow \Omega'$ *and* $\mathsf{dom}(\Delta) = \mathsf{dom}(\Omega')$ *and* $\Omega \longrightarrow \Omega'$
and $\Gamma \vdash e \Rightarrow A'\ p' \dashv \Delta$ *and* $A' = [\Delta]A'$ *and* $A = [\Omega']A'$*.*

(iii) *If* $\Gamma \vdash A\ p$ *type and* $[\Omega]\Gamma \vdash [\Omega]s : [\Omega]A\ p \gg B\ q$ *and* $p' \sqsubseteq p$
then there exist Δ*,* Ω'*,* B' *and* $q' \sqsubseteq q$
such that $\Delta \longrightarrow \Omega'$ *and* $\mathsf{dom}(\Delta) = \mathsf{dom}(\Omega')$ *and* $\Omega \longrightarrow \Omega'$
and $\Gamma \vdash s : [\Gamma]A\ p' \gg B'\ q' \dashv \Delta$ *and* $B' = [\Delta]B'$ *and* $B = [\Omega']B'$*.*

(iv) *If* $\Gamma \vdash A\ p$ *type and* $[\Omega]\Gamma \vdash [\Omega]s : [\Omega]A\ p \gg B\ \lceil q \rceil$ *and* $p' \sqsubseteq p$
then there exist Δ*,* Ω'*,* B'*, and* $q' \sqsubseteq q$
such that $\Delta \longrightarrow \Omega'$ *and* $\mathsf{dom}(\Delta) = \mathsf{dom}(\Omega')$ *and* $\Omega \longrightarrow \Omega'$
and $\Gamma \vdash s : [\Gamma]A\ p' \gg B'\ \lceil q' \rceil \dashv \Delta$ *and* $B' = [\Delta]B'$ *and* $B = [\Omega']B'$*.*

(v) *If* $\Gamma \vdash \vec{A}$ **!** *types and* $\Gamma \vdash C\ p$ *type and* $[\Omega]\Gamma \vdash [\Omega]\Pi :: [\Omega]\vec{A} \Leftarrow [\Omega]C\ p$ *and* $p' \sqsubseteq p$
then there exist Δ*,* Ω'*, and* C
such that $\Delta \longrightarrow \Omega'$ *and* $\mathsf{dom}(\Delta) = \mathsf{dom}(\Omega')$ *and* $\Omega \longrightarrow \Omega'$
and $\Gamma \vdash \Pi :: [\Gamma]\vec{A} \Leftarrow [\Gamma]C\ p' \dashv \Delta$*.*

(vi) *If* $\Gamma \vdash \vec{A}$ **!** *types and* $\Gamma \vdash P$ *prop and* $\mathsf{FEV}(P) = \emptyset$ *and* $\Gamma \vdash C\ p$ *type*
and $[\Omega]\Gamma \;/\; [\Omega]P \vdash [\Omega]\Pi :: [\Omega]\vec{A} \Leftarrow [\Omega]C\ p$
and $p' \sqsubseteq p$
then there exist Δ*,* Ω'*, and* C
such that $\Delta \longrightarrow \Omega'$ *and* $\mathsf{dom}(\Delta) = \mathsf{dom}(\Omega')$ *and* $\Omega \longrightarrow \Omega'$
and $\Gamma \;/\; [\Gamma]P \vdash \Pi :: [\Gamma]\vec{A} \Leftarrow [\Gamma]C\ p' \dashv \Delta$*.*

Proof. By induction, using the measure in Definition 7.

- **Case** $\dfrac{(x : A\ p) \in [\Omega]\Gamma}{[\Omega]\Gamma \vdash x \Rightarrow A\ p}\,\mathsf{DeclVar}$

	$(x : A\ p) \in [\Omega]\Gamma$	Premise
	$\Gamma \longrightarrow \Omega$	Given
	$(x : A'\ p) \in \Gamma$ where $[\Omega]A' = A$	From definition of context application
	Let $\Delta = \Gamma$.	
	Let $\Omega' = \Omega$.	
☞	$\Gamma \longrightarrow \Omega$	Given
☞	$\Omega \longrightarrow \Omega$	By Lemma 32 (Extension Reflexivity)
☞	$\Gamma \vdash x \Rightarrow [\Gamma]A'\ p \dashv \Gamma$	By Var
☞	$[\Gamma]A' = [\Gamma][\Gamma]A'$	By idempotence of substitution
☞	$\mathsf{dom}(\Gamma) = \mathsf{dom}(\Omega)$	Given
	$\Gamma \longrightarrow \Omega$	Given
	$[\Omega][\Gamma]A' = [\Omega]A'$	By Lemma 29 (Substitution Monotonicity) (iii)
☞	$= A$	By above equality

- **Case** $\dfrac{[\Omega]\Gamma \vdash [\Omega]e \Rightarrow \mathrm{B}\ \mathrm{q} \qquad [\Omega]\Gamma \vdash \mathrm{B} \leq^{\pm} [\Omega]A}{[\Omega]\Gamma \vdash [\Omega]e \Leftarrow [\Omega]A\ \mathrm{p}}\ \mathsf{DeclSub}$

	$[\Omega]\Gamma \vdash [\Omega]e \Rightarrow \mathrm{B}\ \mathrm{q}$	Subderivation
	$\Gamma \vdash e \Rightarrow \mathrm{B}'\ \mathrm{q} \dashv \Theta$	By i.h.
	$\mathrm{B} = [\Omega]\mathrm{B}'$	$''$
	$\Theta \longrightarrow \Omega_0$	$''$
	$\Omega \longrightarrow \Omega_0$	$''$
	$\mathsf{dom}(\Theta) = \mathsf{dom}(\Omega_0)$	$''$
	$\Gamma \longrightarrow \Omega$	Given
	$\Gamma \longrightarrow \Omega_0$	By Lemma 33 (Extension Transitivity)
	$[\Omega]\Gamma \vdash \mathrm{B} \leq^{\pm} [\Omega]A$	Subderivation
	$[\Omega]\Gamma = [\Omega]\Theta$	By Lemma 56 (Confluence of Completeness)
	$[\Omega]\Theta \vdash \mathrm{B} \leq^{\pm} [\Omega]A$	By above equalities
	$\Theta \longrightarrow \Omega_0$	Above
	$\Theta \vdash \mathrm{B}' <:^{\pm} A \dashv \Delta$	By Theorem 9 (Completeness of Subtyping)
	$\Omega_0 \longrightarrow \Omega'$	$''$
☞	$\mathsf{dom}(\Delta) = \mathsf{dom}(\Omega')$	$''$
☞	$\Delta \longrightarrow \Omega'$	By Lemma 33 (Extension Transitivity)
☞	$\Omega \longrightarrow \Omega'$	By Lemma 33 (Extension Transitivity)
☞	$\Gamma \vdash e \Leftarrow A\ \mathrm{p} \dashv \Delta$	By Sub

- **Case** $\dfrac{[\Omega]\Gamma \vdash [\Omega]A\ \mathit{type} \qquad [\Omega]\Gamma \vdash [\Omega]e_0 \Leftarrow [\Omega]A\ !}{[\Omega]\Gamma \vdash [\Omega](e_0 : A) \Rightarrow A\ !}\ \mathsf{DeclAnno}$

	$[\Omega]\Gamma \vdash [\Omega]e_0 \Leftarrow [\Omega]A\ !$	Subderivation
	$[\Omega]A = [\Omega][\Gamma]A$	By Lemma 29 (Substitution Monotonicity)
	$[\Omega]\Gamma \vdash [\Omega]e_0 \Leftarrow [\Omega][\Gamma]A\ !$	By above equality
	$\Gamma \vdash e_0 \Leftarrow [\Gamma]A\ ! \dashv \Delta$	By i.h.
☞	$\Delta \longrightarrow \Omega$	$''$
☞	$\Omega \longrightarrow \Omega'$	$''$
☞	$\mathsf{dom}(\Delta) = \mathsf{dom}(\Omega')$	$''$
	$\Delta \longrightarrow \Omega'$	By Lemma 33 (Extension Transitivity)
	$\Gamma \vdash A\ !\ \mathit{type}$	Given
☞	$\Gamma \vdash (e_0 : A) \Rightarrow [\Delta]A\ ! \dashv \Delta$	By Anno
☞	$[\Delta]A = [\Delta][\Delta]A$	By idempotence of substitution
	$A = [\Omega]A$	Above
	$= [\Omega']A$	By Lemma 55 (Completing Completeness) (ii)
☞	$= [\Omega'][\Delta]A$	By Lemma 29 (Substitution Monotonicity)

- **Case**

$$\frac{}{[\Omega]\Gamma \vdash ()\ \Leftarrow 1\ \mathrm{p}}\ \mathsf{Decl1I}$$

We have $[\Omega]A = 1$. Either $[\Gamma]A = 1$, or $[\Gamma]A = \hat{\alpha}$ where $\hat{\alpha} \in \mathsf{unsolved}(\Gamma)$.

In the former case:

	Let $\Delta = \Gamma$.	
	Let $\Omega' = \Omega$.	
☞	$\Gamma \longrightarrow \Omega$	Given
☞	$\Omega \longrightarrow \Omega'$	By Lemma 32 (Extension Reflexivity)
☞	$\mathsf{dom}(\Gamma) = \mathsf{dom}(\Omega)$	Given
	$\Gamma \vdash () \Leftarrow 1\ \mathrm{p} \dashv \Gamma$	By 1I
☞	$\Gamma \vdash () \Leftarrow [\Gamma]1\ \mathrm{p} \dashv \Gamma$	$1 = [\Gamma]1$

In the latter case, since $A = \hat{\alpha}$ and $\Gamma \vdash \hat{\alpha}\ p$ *type* is given, it must be the case that $p = \not{\prime}$.

$\Gamma_0[\hat{\alpha} : \star] \vdash () \Leftarrow \hat{\alpha} \not{\prime} \dashv \Gamma_0[\hat{\alpha} : \star = 1]$ By $1I\hat{\alpha}$
☞ $\Gamma_0[\hat{\alpha} : \star] \vdash () \Leftarrow \big[\Gamma_0[\hat{\alpha} : \star]\big]\hat{\alpha} \not{\prime} \dashv \Gamma_0[\hat{\alpha} : \star = 1]$ By def. of subst.
$\Gamma_0[\hat{\alpha} : \star] \longrightarrow \Omega$ Given
☞ $\Gamma_0[\hat{\alpha} : \star = 1] \longrightarrow \Omega$ By Lemma 27 (Parallel Extension Solution)
☞ $\Omega \longrightarrow \Omega$ By Lemma 32 (Extension Reflexivity)

- **Case** $\dfrac{\nu\ \mathit{chk\text{-}I} \qquad [\Omega]\Gamma, \alpha : \kappa \vdash [\Omega]\nu \Leftarrow A_0\ p}{[\Omega]\Gamma \vdash [\Omega]\nu \Leftarrow \forall\alpha : \kappa.\ A_0\ p}\ \mathsf{Decl}\forall\mathsf{I}$

$[\Omega]A = \forall\alpha : \kappa.\ A_0$ Given
$= \forall\alpha : \kappa.\ [\Omega]A'$ By def. of subst. and predicativity of Ω
$A_0 = [\Omega]A'$ Follows from above equality
$[\Omega]\Gamma, \alpha : \kappa \vdash [\Omega]\nu \Leftarrow [\Omega]A'\ p$ Subderivation and above equality

$\Gamma \longrightarrow \Omega$ Given
$\Gamma, \alpha : \kappa \longrightarrow \Omega, \alpha : \kappa$ By $\longrightarrow$Uvar

$[\Omega]\Gamma, \alpha : \kappa = [\Omega, \alpha : \kappa](\Gamma, \alpha : \kappa)$ By definition of context substitution
$[\Omega, \alpha : \kappa](\Gamma, \alpha : \kappa) \vdash [\Omega]\nu \Leftarrow [\Omega]A'\ p$ By above equality
$[\Omega, \alpha : \kappa](\Gamma, \alpha : \kappa) \vdash [\Omega]\nu \Leftarrow [\Omega, \alpha : \kappa]A'\ p$ By definition of substitution

$\Gamma, \alpha : \kappa \vdash \nu \Leftarrow [\Gamma, \alpha : \kappa]A'\ p \dashv \Delta'$ By i.h.
$\Delta' \longrightarrow \Omega_0'$ ″
$\Omega, \alpha : \kappa \longrightarrow \Omega_0'$ ″
$\mathsf{dom}(\Delta') = \mathsf{dom}(\Omega_0')$ ″
$\Gamma, \alpha : \kappa \longrightarrow \Delta'$ By Lemma 51 (Typing Extension)
$\Delta' = (\Delta, \alpha : \kappa, \Theta)$ By Lemma 22 (Extension Inversion) (i)
$\Delta, \alpha : \kappa, \Theta \longrightarrow \Omega_0'$ By above equality
$\Omega_0' = (\Omega', \alpha : \kappa, \Omega_Z)$ By Lemma 22 (Extension Inversion) (i)
☞ $\Delta \longrightarrow \Omega'$ ″
☞ $\mathsf{dom}(\Delta) = \mathsf{dom}(\Omega')$ ″
☞ $\Omega \longrightarrow \Omega'$ By Lemma 22 (Extension Inversion) on $\Omega, \alpha : \kappa \longrightarrow \Omega_0'$

$\Gamma, \alpha : \kappa \vdash \nu \Leftarrow [\Gamma, \alpha : \kappa]A'\ p \dashv \Delta, \alpha : \kappa, \Theta$ By above equality
$\Gamma, \alpha : \kappa \vdash \nu \Leftarrow [\Gamma]A'\ p \dashv \Delta, \alpha : \kappa, \Theta$ By definition of substitution
$\Gamma \vdash \nu \Leftarrow \forall\alpha : \kappa.\ [\Gamma]A'\ p \dashv \Delta$ By $\forall$I
☞ $\Gamma \vdash \nu \Leftarrow [\Gamma](\forall\alpha : \kappa.\ A')\ p \dashv \Delta$ By definition of substitution

- **Case** $\dfrac{[\Omega]\Gamma \vdash \tau : \kappa \qquad [\Omega]\Gamma \vdash [\Omega](e\ s_0) : [\tau/\alpha][\Omega]A_0 \not{\prime} \gg B\ q}{[\Omega]\Gamma \vdash [\Omega](e\ s_0) : \forall\alpha : \kappa.\ [\Omega]A_0\ p \gg B\ q}\ \mathsf{Decl}\forall\mathsf{Spine}$

$[\Omega]\Gamma \vdash \tau : \kappa$ Subderivation

$\Gamma \longrightarrow \Omega$ Given
$\Gamma, \hat{\alpha} : \kappa \longrightarrow \Omega, \hat{\alpha} : \kappa = \tau$ By $\longrightarrow$Solve

$[\Omega]\Gamma \vdash [\Omega](e\ s_0) : [\tau/\alpha][\Omega]A_0 \not{\prime} \gg B\ q$ Subderivation
$\tau = [\Omega]\tau$ $\mathsf{FEV}(\tau) = \emptyset$
$[\tau/\alpha][\Omega]A_0 = [\tau/\alpha][\Omega, \hat{\alpha} : \kappa = \tau]A_0$ By def. of subst.
$= \big[[\Omega]\tau/\alpha\big]\big[\Omega, \hat{\alpha} : \kappa = \tau\big]A_0$ By above equality
$= \big[\Omega, \hat{\alpha} : \kappa = \tau\big][\hat{\alpha}/\alpha]A_0$ By distributivity of substitution
$[\Omega]\Gamma = [\Omega, \hat{\alpha} : \kappa = \tau](\Gamma, \hat{\alpha} : \kappa)$ By definition of context application

$$
\begin{array}{rl}
[\Omega, \hat{\alpha} : \kappa = \tau](\Gamma, \hat{\alpha} : \kappa) \vdash\ [\Omega](e\ s_0) : \big[\Omega, \hat{\alpha} : \kappa = \tau\big][\hat{\alpha}/\alpha]A_0 \not\gg \mathsf{B}\ q & \text{By above equalities} \\
\Gamma, \hat{\alpha} : \kappa \vdash\ e\ s_0 : [\Gamma, \hat{\alpha} : \kappa][\hat{\alpha}/\alpha]A_0 \not\gg \mathsf{B}'\ q\ \dashv \Delta & \text{By i.h.} \\
\mathsf{B} = [\Omega, \hat{\alpha} : \kappa = \tau]\mathsf{B}' & '' \\
\Delta \longrightarrow \Omega' & '' \\
\mathsf{dom}(\Delta) = \mathsf{dom}(\Omega') & '' \\
\Omega \longrightarrow \Omega' & '' \\
\mathsf{B}' \longrightarrow [\Delta]\mathsf{B}' & '' \\
\mathsf{B} \longrightarrow [\Omega']\mathsf{B}' & ''
\end{array}
$$

$$
\begin{array}{rll}
[\Gamma, \hat{\alpha} : \kappa][\hat{\alpha}/\alpha]A_0 = [\Gamma][\hat{\alpha}/\alpha]A_0 & & \text{By def. of context application} \\
= [\hat{\alpha}/\alpha][\Gamma]A_0 & & \Gamma \text{ does not subst. for } \alpha \\
\Gamma, \hat{\alpha} : \kappa \vdash\ e\ s_0 : [\hat{\alpha}/\alpha][\Gamma]A_0 \not\gg \mathsf{B}'\ q\ \dashv \Delta & & \text{By above equality} \\
\Gamma \vdash\ e\ s_0 : \forall\alpha : \kappa.\ [\Gamma]A_0\ p \gg \mathsf{B}'\ q\ \dashv \Delta & & \text{By } \forall\mathsf{Spine} \\
\Gamma \vdash\ e\ s_0 : [\Gamma](\forall\alpha : \kappa.\ A_0)\ p \gg \mathsf{B}'\ q\ \dashv \Delta & & \text{By def. of subst.}
\end{array}
$$

- **Case** $\dfrac{\nu\ \mathit{chk\text{-}I} \qquad [\Omega]\Gamma\ /\ [\Omega]P \vdash\ [\Omega]\nu \Leftarrow [\Omega]A_0\ !}{[\Omega]\Gamma \vdash\ [\Omega]\nu \Leftarrow ([\Omega]P) \supset [\Omega]A_0\ !}\ \mathsf{DeclI}\supset$

  $[\Omega]\Gamma\ /\ [\Omega]P \vdash\ [\Omega]\nu \Leftarrow [\Omega]A_0\ !$ Subderivation

  The concluding rule in this subderivation must be DeclCheck$\perp$ or DeclCheckUnify. In either case, $[\Omega]P$ has the form $(\sigma' = \tau')$ where $\sigma' = [\Omega]\sigma$ and $\tau' = [\Omega]\tau$.

  - **Case** $\dfrac{\mathsf{mgu}([\Omega]\sigma, [\Omega]\tau) = \bot}{[\Omega]\Gamma\ /\ [\Omega](\sigma = \tau) \vdash\ [\Omega]\nu \Leftarrow [\Omega]A_0\ !}\ \mathsf{DeclCheck}\bot$

    We have $\mathsf{mgu}([\Omega]\sigma, [\Omega]\tau) = \bot$. To apply Lemma 92 (Completeness of Elimeq) (2), we need to show conditions 1–5.

    $$
    \begin{array}{rll}
    *** & \Gamma \vdash (\sigma = \tau) \supset A_0\ !\ \mathit{type} & \text{Given} \\
    & [\Omega]\big((\sigma{=}\tau) \supset A_0\big) = [\Gamma]\big((\sigma{=}\tau) \supset A_0\big) & \text{By Lemma 39 (Principal Agreement) (i)} \\
    & [\Omega]\sigma = [\Gamma]\sigma & \text{By a property of subst.} \\
    & [\Omega]\tau = [\Gamma]\tau & \text{Similar}
    \end{array}
    $$

    $$
    \begin{array}{rll}
    & \Gamma \vdash \sigma : \kappa & \text{By inversion} \\
    3 & \Gamma \vdash [\Gamma]\sigma : \kappa & \text{By Lemma 11 (Right-Hand Substitution for Sorting)} \\
    4 & \Gamma \vdash [\Gamma]\tau : \kappa & \text{Similar} \\
    & \mathsf{mgu}([\Omega]\sigma, [\Omega]\tau) = \bot & \text{Given} \\
    & \mathsf{mgu}([\Gamma]\sigma, [\Gamma]\tau) = \bot & \text{By above equalities}
    \end{array}
    $$

    $$
    \begin{array}{rll}
    & \mathsf{FEV}(\sigma) \cup \mathsf{FEV}(\tau) = \emptyset & \text{By inversion on } *** \\
    & \mathsf{FEV}([\Omega]\sigma) \cup \mathsf{FEV}([\Omega]\tau) = \emptyset & \text{By a property of complete contexts} \\
    5 & \mathsf{FEV}([\Gamma]\sigma) \cup \mathsf{FEV}([\Gamma]\tau) = \emptyset & \text{By above equalities} \\
    1 & [\Gamma][\Gamma]\sigma = [\Gamma]\sigma & \text{By idempotence of subst.} \\
    2 & [\Gamma][\Gamma]\tau = [\Gamma]\tau & \text{By idempotence of subst.}
    \end{array}
    $$

    $$
    \begin{array}{rll}
    \Gamma\ /\ [\Gamma]\sigma \stackrel{\circ}{=} [\Gamma]\tau : \kappa\ \dashv \bot & & \text{By Lemma 92 (Completeness of Elimeq) (2)} \\
    \Gamma, \blacktriangleright_P\ /\ [\Gamma]\sigma = [\Gamma]\tau\ \dashv \bot & & \text{By ElimpropEq}
    \end{array}
    $$

    $$
    \begin{array}{rll}
    & \Gamma \vdash \nu \Leftarrow ([\Gamma]\sigma = [\Gamma]\tau) \supset [\Gamma]A_0\ !\ \dashv \Gamma & \text{By } \supset\mathsf{I}\bot \\
    \text{☞} & \Gamma \vdash \nu \Leftarrow [\Gamma]\big((\sigma = \tau) \supset A_0\big)\ !\ \dashv \Gamma & \text{By def. of subst.} \\
    \text{☞} & \Gamma \longrightarrow \Omega & \text{Given} \\
    \text{☞} & \Omega \longrightarrow \Omega & \text{By Lemma 32 (Extension Reflexivity)} \\
    \text{☞} & \mathsf{dom}(\Gamma) = \mathsf{dom}(\Omega) & \text{Given}
    \end{array}
    $$

  - **Case** $\dfrac{\mathsf{mgu}([\Omega]\sigma, [\Omega]\tau) = \theta \qquad \theta([\Omega]\Gamma) \vdash\ \theta([\Omega]e) \Leftarrow \theta([\Omega]A_0)\ !}{[\Omega]\Gamma\ /\ (([\Omega]\sigma) = [\Omega]\tau) \vdash\ [\Omega]e \Leftarrow [\Omega]A_0\ !}\ \mathsf{DeclCheckUnify}$

We have $\mathsf{mgu}([\Omega]\sigma, [\Omega]\tau) = \theta$, and will need to apply Lemma 92 (Completeness of Elimeq) (1). That lemma has five side conditions, which can be shown exactly as in the DeclCheck$\perp$ case above.

$\mathsf{mgu}(\sigma,\tau) = \theta$ Premise
Let $\Omega_0 = (\Omega, \blacktriangleright_P)$.
$\Gamma \longrightarrow \Omega$ Given
$\Gamma, \blacktriangleright_P \longrightarrow \Omega_0$ By $\longrightarrow$Marker

$\mathsf{dom}(\Gamma) = \mathsf{dom}(\Omega)$ Given
$\mathsf{dom}(\Gamma, \blacktriangleright_P) = \mathsf{dom}(\Omega_0)$ By def. of $\mathsf{dom}(-)$
$\Gamma, \blacktriangleright_P \ / \ [\Gamma]\sigma \stackrel{\circ}{=} [\Gamma]\tau : \kappa \dashv \Gamma, \blacktriangleright_P, \Theta$ By Lemma 92 (Completeness of Elimeq) (1)

$\Gamma, \blacktriangleright_P \ / \ [\Gamma]\sigma = [\Gamma]\tau \dashv \Gamma, \blacktriangleright_P, \Theta$ By ElimpropEq
EQ0 for all $\Gamma, \blacktriangleright_P \vdash u : \kappa.\ [\Gamma, \blacktriangleright_P, \Theta]u = \theta([\Gamma, \blacktriangleright_P]u)$ $''$

$\Gamma \vdash P \supset A_0\ \mathbf{!}\ \mathit{type}$ Given
$\Gamma \vdash A_0\ \mathbf{!}\ \mathit{type}$ By inversion
$\Gamma \longrightarrow \Omega$ Given
EQa $[\Gamma]A_0 = [\Omega]A_0$ By Lemma 39 (Principal Agreement) (i)

Let $\Omega_1 = (\Omega, \blacktriangleright_P, \Theta)$.
$\theta([\Omega]\Gamma) \vdash \theta(e) \Leftarrow \theta([\Omega]A_0)\ \mathbf{!}$ Subderivation

$\Gamma, \blacktriangleright_P, \Theta \longrightarrow \Omega_1$ By induction on Θ

$\theta([\Omega]A_0) = \theta([\Gamma]A_0)$ By above equality EQa
$= [\Gamma, \blacktriangleright_P, \Theta]A_0$ By Lemma 93 (Substitution Upgrade) (i) (with EQ0)
$= [\Omega_1]A_0$ By Lemma 39 (Principal Agreement) (i)
$= [\Omega_1][\Gamma, \blacktriangleright_P, \Theta]A_0$ By Lemma 29 (Substitution Monotonicity) (iii)

$\theta([\Omega]\Gamma) = [\Omega_1](\Gamma, \blacktriangleright_P, \Theta)$ By Lemma 93 (Substitution Upgrade) (iii)
$\theta([\Omega]e) = [\Omega_1]e$ By Lemma 93 (Substitution Upgrade) (iv)

$[\Omega_1](\Gamma, \blacktriangleright_P, \Theta) \vdash [\Omega_1]e \Leftarrow [\Omega_1][\Gamma, \blacktriangleright_P, \Theta]A_0\ \mathbf{!}$ By above equalities

$\mathsf{dom}(\Gamma, \blacktriangleright_P, \Theta) = \mathsf{dom}(\Omega_1)$ $\mathsf{dom}(\Gamma) = \mathsf{dom}(\Omega)$

$\Gamma, \blacktriangleright_P, \Theta \vdash e \Leftarrow [\Gamma, \blacktriangleright_P, \Theta]A_0\ \mathbf{!} \dashv \Delta'$ By i.h.
$\Delta' \longrightarrow \Omega_2'$ $''$
$\Omega_1 \longrightarrow \Omega_2'$ $''$
$\mathsf{dom}(\Delta') = \mathsf{dom}(\Omega_2')$ $''$
$\Delta' = (\Delta, \blacktriangleright_P, \Delta'')$ By Lemma 22 (Extension Inversion) (ii)
$\Omega_2' = (\Omega', \blacktriangleright_P, \Omega_Z)$ By Lemma 22 (Extension Inversion) (ii)
☞ $\Delta \longrightarrow \Omega'$ $''$
$\Omega_0 \longrightarrow \Omega_2'$ By Lemma 33 (Extension Transitivity)
$\Omega, \blacktriangleright_P \longrightarrow \Omega', \blacktriangleright_P, \Omega_Z$ By above equalities
☞ $\Omega \longrightarrow \Omega'$ By Lemma 22 (Extension Inversion) (ii)
☞ $\mathsf{dom}(\Delta) = \mathsf{dom}(\Omega')$ $''$

$\Gamma, \blacktriangleright_P, \Theta \vdash e \Leftarrow [\Gamma, \blacktriangleright_P, \Theta]A_0\ \mathbf{!} \dashv \Delta, \blacktriangleright_P, \Delta''$ By above equality
$\Gamma \vdash e \Leftarrow ([\Gamma]\sigma = [\Gamma]\tau) \supset [\Gamma]A_0\ \mathbf{!} \dashv \Delta$ By $\supset$I
☞ $\Gamma \vdash e \Leftarrow [\Gamma](P \supset A_0)\ \mathbf{!} \dashv \Delta$ By def. of subst.

- **Case**
$$\frac{[\Omega]\Gamma \vdash [\Omega]P\ \mathit{true} \qquad [\Omega]\Gamma \vdash [\Omega](e\ s_0) : [\Omega]A_0\ p \gg B\ q}{[\Omega]\Gamma \vdash [\Omega](e\ s_0) : ([\Omega]P) \supset [\Omega]A_0\ p \gg B\ q}\ \mathsf{Decl}\supset\mathsf{Spine}$$

$[\Omega]\Gamma \vdash [\Omega]P$ *true* — Subderivation
$[\Omega]\Gamma \vdash [\Omega][\Gamma]P$ *true* — By Lemma 29 (Substitution Monotonicity) (ii)
$\Gamma \vdash [\Gamma]P$ *true* $\dashv \Theta$ — By Lemma 95 (Completeness of Checkprop)
$\Theta \longrightarrow \Omega_1$ — ″
$\Omega \longrightarrow \Omega_1$ — ″
$\mathsf{dom}(\Theta) = \mathsf{dom}(\Omega_1)$ — ″

$\Gamma \longrightarrow \Omega$ — Given
$[\Omega]\Gamma = [\Omega_1]\Theta$ — By Lemma 57 (Multiple Confluence)
$[\Omega]A_0 = [\Omega_1]A_0$ — By Lemma 55 (Completing Completeness) (ii)
$[\Omega]\Gamma \vdash [\Omega](e\ s_0) : [\Omega]A_0\ p \gg B\ q$ — Subderivation
$[\Omega_1]\Theta \vdash [\Omega](e\ s_0) : [\Omega_1]A_0\ p \gg B\ q$ — By above equalities
$\Theta \vdash e\ s_0 : [\Theta]A_0\ p \gg B'\ q \dashv \Delta$ — By i.h.
☞ $B' = [\Delta]B'$ — ″
☞ $\mathsf{dom}(\Delta) = \mathsf{dom}(\Omega')$ — ″
☞ $B = [\Omega']B'$ — ″
☞ $\Delta \longrightarrow \Omega'$ — ″
$\Omega_1 \longrightarrow \Omega'$ — ″
☞ $\Omega \longrightarrow \Omega'$ — By Lemma 33 (Extension Transitivity)
$[\Theta]A_0 = [\Theta][\Gamma]A_0$ — By Lemma 29 (Substitution Monotonicity) (iii)

$\Theta \vdash e\ s_0 : [\Theta][\Gamma]A_0\ p \gg B'\ q \dashv \Delta$ — By above equality
$\Gamma \vdash e\ s_0 : ([\Gamma]P) \supset [\Gamma]A_0\ p \gg B'\ q \dashv \Delta$ — By $\supset$Spine
☞ $\Gamma \vdash e\ s_0 : [\Gamma](P \supset A_0)\ p \gg B'\ q \dashv \Delta$ — By def. of subst.

- **Case**
$$\frac{[\Omega]\Gamma \vdash [\Omega]e_0 \Leftarrow A'_k\ p}{[\Omega]\Gamma \vdash \mathsf{inj}_k\ [\Omega]e_0 \Leftarrow \underbrace{A'_1 + A'_2}_{[\Omega]A}\ p}\ \mathsf{Decl}{+}\mathsf{I}_k$$

Either $[\Gamma]A = A_1 + A_2$ (where $[\Omega]A_k = A'_k$) or $[\Gamma]A = \hat{\alpha} \in \mathsf{unsolved}(\Gamma)$.

In the former case:

$[\Omega]\Gamma \vdash [\Omega]e_0 \Leftarrow A'_k\ p$ — Subderivation
$[\Omega]\Gamma \vdash [\Omega]e_0 \Leftarrow [\Omega]A_k\ p$ — $[\Omega]A_k = A'_k$
$\Gamma \vdash e_0 \Leftarrow [\Gamma]A_k\ p \dashv \Delta$ — By i.h.
☞ $\Delta \longrightarrow \Omega$ — ″
☞ $\mathsf{dom}(\Delta) = \mathsf{dom}(\Omega')$ — ″
☞ $\Omega \longrightarrow \Omega'$ — ″
$\Gamma \vdash \mathsf{inj}_k\ e_0 \Leftarrow ([\Gamma]A_1) + ([\Gamma]A_2)\ p \dashv \Delta$ — By $+\mathsf{I}_k$
☞ $\Gamma \vdash \mathsf{inj}_k\ e_0 \Leftarrow [\Gamma](A_1 + A_2)\ p \dashv \Delta$ — By def. of subst.

In the latter case, $A = \hat{\alpha}$ and $[\Omega]A = [\Omega]\hat{\alpha} = A'_1 + A'_2 = \tau'_1 + \tau'_2$.
By inversion on $\Gamma \vdash \hat{\alpha}\ p$ *type*, it must be the case that $p = \not{\prime}$.

$\Gamma \longrightarrow \Omega$ — Given
$\Gamma = \Gamma_0[\hat{\alpha} : \star]$ — $\hat{\alpha} \in \mathsf{unsolved}(\Gamma)$
$\Omega = \Omega_0[\hat{\alpha} : \star = \tau_0]$ — By Lemma 22 (Extension Inversion) (vi)
Let $\Omega_2 = \Omega_0[\hat{\alpha}_1 : \star = \tau'_1, \hat{\alpha}_1 : \star = \tau'_2, \hat{\alpha} : \star = \hat{\alpha}_1 + \hat{\alpha}_2]$.
Let $\Gamma_2 = \Gamma_0[\hat{\alpha}_1 : \star, \hat{\alpha}_2 : \star, \hat{\alpha} : \star = \hat{\alpha}_1 + \hat{\alpha}_2]$.

$\Gamma \longrightarrow \Gamma_2$ — By Lemma 23 (Deep Evar Introduction) (iii) twice and Lemma 26 (Parallel Admissibility) (ii)
$\Omega \longrightarrow \Omega_2$ — By Lemma 23 (Deep Evar Introduction) (iii) twice and Lemma 26 (Parallel Admissibility) (iii)
$\Gamma_2 \longrightarrow \Omega_2$ — By Lemma 26 (Parallel Admissibility) (ii), (ii), (iii)

$$
\begin{array}{rll}
[\Omega]\Gamma \vdash [\Omega]e_0 \Leftarrow [\Omega_2]\hat{\alpha}_k \not\!/\!/ & & \text{Subd. and } A'_k = \tau'_k = [\Omega_2]\hat{\alpha}_k \\
[\Omega]\Gamma = [\Omega_2]\Gamma_2 & & \text{By Lemma 57 (Multiple Confluence)} \\
[\Omega_2]\Gamma_2 \vdash e_0 \Leftarrow [\Omega_2]\hat{\alpha}_k \not\!/\!/ & & \text{By above equality} \\
\Gamma_2 \vdash e_0 \Leftarrow [\Gamma_2]\hat{\alpha}_k \not\!/\!/ \dashv \Delta & & \text{By i.h.} \\
\Delta \longrightarrow \Omega' & & '' \\
\mathsf{dom}(\Delta) = \mathsf{dom}(\Omega') & & '' \\
\Omega_2 \longrightarrow \Omega' & & '' \\
\Omega \longrightarrow \Omega' & & \text{By Lemma 33 (Extension Transitivity)} \\
\Gamma \vdash \mathsf{inj}_k\, e_0 \Rightarrow \hat{\alpha} \not\!/\!/ \dashv \Delta & & \text{By } +\mathsf{I}\hat{\alpha}_k \\
\Gamma \vdash \mathsf{inj}_k\, e_0 \Rightarrow [\Gamma]\hat{\alpha} \not\!/\!/ \dashv \Delta & & \hat{\alpha} \in \mathsf{unsolved}(\Gamma)
\end{array}
$$

- **Case** $\dfrac{[\Omega]\Gamma, x : A'_1\ p \vdash [\Omega]e_0 \Leftarrow A'_2\ p}{[\Omega]\Gamma \vdash \lambda x.\, [\Omega]e_0 \Leftarrow A'_1 \to A'_2\ p}\ \mathsf{Decl}{\to}\mathsf{I}$

  We have $[\Omega]A = A'_1 \to A'_2$. Either $[\Gamma]A = A_1 \to A_2$ where $A'_1 = [\Omega]A_1$ and $A'_2 = [\Omega]A_2$—or $[\Gamma]A = \hat{\alpha}$ and $[\Omega]\hat{\alpha} = A'_1 \to A'_2$.

  In the former case:

  $$
  \begin{array}{rll}
  [\Omega]\Gamma, x : A'_1\ p \vdash [\Omega]e_0 \Leftarrow A'_2\ p & & \text{Subderivation} \\
  \\
  A'_1 = [\Omega]A_1 & & \text{Known in this subcase} \\
  = [\Omega][\Gamma]A_1 & & \text{By Lemma 30 (Substitution Invariance)} \\
  [\Omega]A'_1 = [\Omega][\Omega][\Gamma]A_1 & & \text{Applying } \Omega \text{ on both sides} \\
  = [\Omega][\Gamma]A_1 & & \text{By idempotence of substitution} \\
  \\
  [\Omega]\Gamma, x : A'_1\ p = [\Omega, x : A'_1\ p](\Gamma, x : [\Gamma]A_1\ p) & & \text{By definition of context application} \\
  [\Omega, x : A'_1\ p](\Gamma, x : [\Gamma]A_1\ p) \vdash [\Omega]e_0 \Leftarrow A'_2\ p & & \text{By above equality} \\
  \\
  \Gamma \longrightarrow \Omega & & \text{Given} \\
  \Gamma, x : [\Gamma]A_1\ p \longrightarrow \Omega, x : A'_1\ p & & \text{By } \longrightarrow\mathsf{Var} \\
  \\
  \mathsf{dom}(\Gamma) = \mathsf{dom}(\Omega) & & \text{Given} \\
  \mathsf{dom}(\Gamma, x : [\Gamma]A_1\ p) = \Omega, x : A'_1\ p & & \text{By def. of } \mathsf{dom}(-) \\
  \Gamma, x : [\Gamma]A_1\ p \vdash e_0 \Leftarrow A_2\ p \dashv \Delta' & & \text{By i.h.} \\
  \Delta' \longrightarrow \Omega'_0 & & '' \\
  \mathsf{dom}(\Delta') = \mathsf{dom}(\Omega'_0) & & '' \\
  \Omega, x : A'_1\ p \longrightarrow \Omega'_0 & & '' \\
  \Omega'_0 = (\Omega', x : A'_1\ p, \Omega_Z) & & \text{By Lemma 22 (Extension Inversion) (v)} \\
  \Omega \longrightarrow \Omega' & & '' \\
  \\
  \Gamma, x : [\Gamma]A_1\ p \longrightarrow \Delta' & & \text{By Lemma 51 (Typing Extension)} \\
  \Delta' = (\Delta, x : \cdots, \Theta) & & \text{By Lemma 22 (Extension Inversion) (v)} \\
  \Delta, x : \cdots, \Theta \longrightarrow \Omega', x : A'_1\ p, \Omega_Z & & \text{By above equalities} \\
  \Delta \longrightarrow \Omega' & & \text{By Lemma 22 (Extension Inversion) (v)} \\
  \mathsf{dom}(\Delta) = \mathsf{dom}(\Omega') & & '' \\
  \\
  \Gamma, x : [\Gamma]A_1\ p \vdash e_0 \Leftarrow [\Gamma]A_2\ p \dashv \Delta, x : \cdots p, \Theta & & \text{By above equality} \\
  \Gamma \vdash \lambda x.\, e_0 \Leftarrow ([\Gamma]A_1) \to ([\Gamma]A_2)\ p \dashv \Delta & & \text{By } \to\mathsf{I} \\
  \Gamma \vdash \lambda x.\, e_0 \Leftarrow [\Gamma](A_1 \to A_2)\ p \dashv \Delta & & \text{By definition of substitution}
  \end{array}
  $$

  In the latter case ($[\Gamma]A = \hat{\alpha} \in \mathsf{unsolved}(\Gamma)$ and $[\Omega]\hat{\alpha} = A'_1 \to A'_2 = \tau'_1 \to \tau'_2$):

  By inversion on $\Gamma \vdash \hat{\alpha}\ p\ \mathit{type}$, it must be the case that $p = \not\!/\!/$.
  Since $\hat{\alpha} \in \mathsf{unsolved}(\Gamma)$, the context Γ must have the form $\Gamma_0[\hat{\alpha} : \star]$.
  Let $\Gamma_2 = \Gamma_0[\hat{\alpha}_1 : \star, \hat{\alpha}_2 : \star, \hat{\alpha} : \star = \hat{\alpha}_1 \to \hat{\alpha}_2]$.

$\Gamma \longrightarrow \Gamma_2$ By Lemma 23 (Deep Evar Introduction) (iii) twice and Lemma 26 (Parallel Admissibility) (ii)

$[\Omega]\hat{\alpha} = \tau_1' \to \tau_2'$ Known in this subcase

$\Gamma \longrightarrow \Omega$ Given

$\Omega = \Omega_0[\hat{\alpha} : \star = \tau_0]$ By Lemma 22 (Extension Inversion) (vi)

Let $\Omega_2 = \Omega_0[\hat{\alpha}_1 : \star = \tau_1', \hat{\alpha}_1 : \star = \tau_2', \hat{\alpha} : \star = \hat{\alpha}_1 \to \hat{\alpha}_2]$.

$\Gamma \longrightarrow \Gamma_2$ By Lemma 23 (Deep Evar Introduction) (iii) twice and Lemma 26 (Parallel Admissibility) (ii)

$\Omega \longrightarrow \Omega_2$ By Lemma 23 (Deep Evar Introduction) (iii) twice and Lemma 26 (Parallel Admissibility) (iii)

$\Gamma_2 \longrightarrow \Omega_2$ By Lemma 26 (Parallel Admissibility) (ii), (ii), (iii)

$[\Omega]\Gamma, x : \tau_1' \not\parallel \vdash [\Omega]e_0 \Leftarrow \tau_2' \not\parallel$ Subderivation

$[\Omega]\Gamma = [\Omega_2]\Gamma_2$ By Lemma 57 (Multiple Confluence)

$\tau_2' = [\Omega]\hat{\alpha}_2$ From above equality

$= [\Omega_2]\hat{\alpha}_2$ By Lemma 55 (Completing Completeness) (i)

$\tau_1' = [\Omega_2]\hat{\alpha}_1$ Similar

$[\Omega_2]\Gamma_2, x : \tau_1' \not\parallel = [\Omega_2, x : \tau_1' \not\parallel](\Gamma_2, x : \hat{\alpha}_1 \not\parallel)$ By def. of context application

$[\Omega_2, x : \tau_1' \not\parallel](\Gamma_2, x : \hat{\alpha}_1 \not\parallel) \vdash [\Omega]e_0 \Leftarrow [\Omega_2]\hat{\alpha}_2 \not\parallel$ By above equalities

$\mathsf{dom}(\Gamma) = \mathsf{dom}(\Omega)$ Given

$\mathsf{dom}(\Gamma_2, x : \hat{\alpha}_1 \not\parallel) = \mathsf{dom}(\Omega_2, x : \tau_1' \not\parallel)$ By def. of Γ_2 and Ω_2

$\Gamma_2, x : \hat{\alpha}_1 \not\parallel \vdash e_0 \Leftarrow [\Gamma_2, x : \hat{\alpha}_1 \not\parallel]\hat{\alpha}_2 \not\parallel \dashv \Delta^+$ By i.h.

$\Delta^+ \longrightarrow \Omega^+$ ″

$\mathsf{dom}(\Delta^+) = \mathsf{dom}(\Omega^+)$ ″

$\Omega_2 \longrightarrow \Omega^+$ ″

$\Gamma_2, x : \hat{\alpha}_1 \not\parallel \longrightarrow \Delta^+$ By Lemma 51 (Typing Extension)

$\Delta^+ = (\Delta, x : \hat{\alpha}_1 \not\parallel, \Delta_Z)$ By Lemma 22 (Extension Inversion) (v)

$\Omega^+ = (\Omega', x : \ldots \not\parallel, \Omega_Z)$ By Lemma 22 (Extension Inversion) (v)

☞ $\Delta \longrightarrow \Omega'$ ″

☞ $\mathsf{dom}(\Delta) = \mathsf{dom}(\Omega')$ ″

$\Omega \longrightarrow \Omega_2$ Above

$\Omega \longrightarrow \Omega^+$ By Lemma 33 (Extension Transitivity)

☞ $\Omega \longrightarrow \Omega'$ By Lemma 22 (Extension Inversion) (v)

$\Gamma \vdash \lambda x.\, e_0 \Leftarrow \hat{\alpha} \not\parallel \dashv \Delta$ By $\to$I$\hat{\alpha}$

$\hat{\alpha} = [\Gamma]\hat{\alpha}$ $\hat{\alpha} \in \mathsf{unsolved}(\Gamma)$

☞ $\Gamma \vdash \lambda x.\, e_0 \Leftarrow [\Gamma]\hat{\alpha} \not\parallel \dashv \Delta$ By above equality

- **Case** $\dfrac{[\Omega]\Gamma, x : [\Omega]A\ p \vdash\ [\Omega]v \Leftarrow [\Omega]A\ p}{[\Omega]\Gamma \vdash\ \mathsf{rec}\ x.\ [\Omega]v \Leftarrow [\Omega]A\ p}\ \mathsf{DeclRec}$

$[\Omega]\Gamma, x : [\Omega]A\ p \vdash [\Omega]v \Leftarrow [\Omega]A\ p$ Subderivation

$[\Omega]\Gamma, x : [\Omega]A\ p = [\Omega, x : [\Omega]A\ p](\Gamma, x : [\Gamma]A\ p)$ By definition of context application

$[\Omega, x : [\Omega]A\ p](\Gamma, x : [\Gamma]A\ p) \vdash [\Omega]v \Leftarrow [\Omega]A\ p$ By above equality

$\Gamma \longrightarrow \Omega$ Given

$\Gamma, x : [\Gamma]A\ p \longrightarrow \Omega, x : [\Omega]A\ p$ By $\longrightarrow$Var

$\mathsf{dom}(\Gamma) = \mathsf{dom}(\Omega)$ Given

$\mathsf{dom}(\Gamma, x : [\Gamma]A\ p) = \Omega, x : [\Omega]A\ p$ By def. of $\mathsf{dom}(-)$

$\Gamma, x : [\Gamma]A\ p \vdash v \Leftarrow [\Gamma]A\ p\ \dashv \Delta'$ By i.h.

$\Delta' \longrightarrow \Omega'_0$ ″

$\mathsf{dom}(\Delta') = \mathsf{dom}(\Omega'_0)$ ″

$\Omega, x : [\Omega]A\ p \longrightarrow \Omega'_0$ ″

$\Omega'_0 = (\Omega', x : [\Omega]A\ p, \Theta)$ By Lemma 22 (Extension Inversion) (v)

☞ $\Omega \longrightarrow \Omega'$ ″

$\Gamma, x : [\Gamma]A\ p \longrightarrow \Delta'$ By Lemma 51 (Typing Extension)

$\Delta' = (\Delta, x : \cdots, \Theta)$ By Lemma 22 (Extension Inversion) (v)

$\Delta, x : \cdots, \Theta \longrightarrow \Omega', x : [\Omega]A\ p, \Theta$ By above equalities

☞ $\Delta \longrightarrow \Omega'$ By Lemma 22 (Extension Inversion) (v)

☞ $\mathsf{dom}(\Delta) = \mathsf{dom}(\Omega')$ ″

$\Gamma, x : [\Gamma]A\ p \vdash v \Leftarrow [\Gamma]A\ p\ \dashv \Delta, x : [\Gamma]A\ p, \Theta$ By above equality

☞ $\Gamma \vdash \mathsf{rec}\ x.v \Leftarrow [\Gamma]A\ p\ \dashv \Delta$ By Rec

- **Case** $\dfrac{[\Omega]\Gamma \vdash\ [\Omega]e_0 \Rightarrow A\ q \qquad [\Omega]\Gamma \vdash\ [\Omega]s_0 : A\ q \gg C\ \lceil p \rceil}{[\Omega]\Gamma \vdash\ [\Omega](e_0\ s_0) \Rightarrow C\ p}\ \mathsf{Decl{\to}E}$

$[\Omega]\Gamma \vdash [\Omega]e_0 \Rightarrow A\ q$ Subderivation

$\Gamma \vdash e_0 \Rightarrow A'\ q\ \dashv \Theta$ By i.h.

$\Theta \longrightarrow \Omega_\Theta$ ″

$\mathsf{dom}(\Theta) = \mathsf{dom}(\Omega_\Theta)$ ″

$\Omega \longrightarrow \Omega_\Theta$ ″

$A = [\Omega_\Theta]A'$ ″

$A' = [\Theta]A'$ ″

$\Gamma \longrightarrow \Omega$ Given

$[\Omega]\Gamma = [\Omega_\Theta]\Theta$ By Lemma 57 (Multiple Confluence)

$[\Omega]\Gamma \vdash\ [\Omega]s_0 : A\ q \gg C\ \lceil p \rceil$ Subderivation

$[\Omega_\Theta]\Theta \vdash\ [\Omega]s_0 : [\Omega_\Theta]A'\ q \gg C\ \lceil p \rceil$ By above equalities

$\Theta \vdash\ s_0 : [\Theta]A'\ q \gg C'\ \lceil p \rceil\ \dashv \Delta$ By i.h.

☞ $C' = [\Delta]C'$ ″

☞ $\Delta \longrightarrow \Omega'$ ″

☞ $\mathsf{dom}(\Delta) = \mathsf{dom}(\Omega')$ ″

$\Omega_\Theta \longrightarrow \Omega'$ ″

☞ $C = [\Omega']C'$ ″

$\Theta \vdash\ s_0 : A'\ q \gg C'\ \lceil p \rceil\ \dashv \Delta$ By above equality

☞ $\Omega \longrightarrow \Omega'$ By Lemma 33 (Extension Transitivity)

☞ $\Gamma \vdash e_0\ s_0 \Rightarrow C'\ p\ \dashv \Delta$ By $\to$E

- **Case**

$$\frac{[\Omega]\Gamma \vdash [\Omega]s : [\Omega]A\ \mathbf{!} \gg C\ \not\Downarrow \qquad \text{for all } C_2.\ \text{if } [\Omega]\Gamma \vdash [\Omega]s : [\Omega]A\ \mathbf{!} \gg C_2\ \not\Downarrow \text{ then } C_2 = C}{[\Omega]\Gamma \vdash [\Omega]s : [\Omega]A\ \mathbf{!} \gg C\ \lceil\mathbf{!}\rceil}\ \mathsf{DeclSpineRecover}$$

	$\Gamma \longrightarrow \Omega$	Given
	$[\Omega]\Gamma \vdash [\Omega]s : [\Omega]A\ \mathbf{!} \gg C\ \not\Downarrow$	Subderivation
	$\Gamma \vdash s : [\Gamma]A\ \mathbf{!} \gg C'\ \not\Downarrow \dashv \Delta$	By i.h.
☞	$\Delta \longrightarrow \Omega'$	$''$
☞	$\Omega \longrightarrow \Omega'$	$''$
☞	$\mathsf{dom}(\Delta) = \mathsf{dom}(\Omega')$	$''$
☞	$C = [\Omega']C'$	$''$
☞	$C' = [\Delta]C'$	$''$

Suppose, for a contradiction, that $\mathsf{FEV}([\Delta]C') \neq \emptyset$.
That is, there exists some $\hat{\alpha} \in \mathsf{FEV}([\Delta]C')$.

	$\Delta \longrightarrow \Omega_2$	By Lemma 60 (Split Solutions)
	$\underbrace{\Omega_1'[\hat{\alpha} : \kappa = t_1]}_{\Omega_1} \longrightarrow \Omega'$	$''$
	$\Omega_2 = \Omega_1'[\hat{\alpha} : \kappa = t_2]$	$''$
	$t_2 \neq t_1$	$''$
(NEQ)	$[\Omega_2]\hat{\alpha} \neq [\Omega_1']\hat{\alpha}$	By def. of subst. ($t_2 \neq t_1$)
(EQ)	$[\Omega_2]\hat{\beta} = [\Omega_1']\hat{\beta}$ for all $\hat{\beta} \neq \hat{\alpha}$	By construction of Ω_2 and Ω_2 canonical

Choose $\hat{\alpha}_R$ such that $\hat{\alpha}_R \in \mathsf{FEV}(C')$ and either $\hat{\alpha}_R = \hat{\alpha}$ or $\hat{\alpha} \in \mathsf{FEV}([\Delta]\hat{\alpha}_R)$.
Then either $\hat{\alpha}_R = \hat{\alpha}$, or $\hat{\alpha}_R$ is declared to the right of $\hat{\alpha}$ in Δ.

$[\Omega_2]C' \neq [\Omega']C'$	From (NEQ) and (EQ)
$\Gamma \vdash s : [\Gamma]A\ \mathbf{!} \gg C'\ \not\Downarrow \dashv \Delta$	Above
$[\Omega_2]\Gamma \vdash [\Omega_2]s : [\Omega_2][\Gamma]A\ \mathbf{!} \gg [\Omega_2]C'\ \not\Downarrow$	By Theorem 8 (Soundness of Algorithmic Typing)
$\Gamma \vdash s : [\Gamma]A\ \mathbf{!} \gg C'\ \not\Downarrow \dashv \Delta$	Above
$\Gamma \vdash A\ \mathbf{!}\ \mathit{type}$	Given
$\Gamma \vdash [\Gamma]A\ \mathbf{!}\ \mathit{type}$	By Lemma 13 (Right-Hand Substitution for Typing)
$\mathsf{FEV}([\Gamma]A) = \emptyset$	By inversion
$\mathsf{FEV}([\Gamma]A) \subseteq \mathsf{dom}(\cdot)$	Property of $\subseteq$
$\Delta = (\Delta_L * \Delta_R)$	By Lemma 72 (Separation—Main) (Spines)
$(\Gamma * \cdot) \xrightarrow[*]{} (\Delta_L * \Delta_R)$	$''$
$\mathsf{FEV}(C') \subseteq \mathsf{dom}(\Delta_R)$	$''$
$\hat{\alpha}_R \in \mathsf{FEV}(C')$	Above
$\hat{\alpha}_R \in \mathsf{dom}(\Delta_R)$	Property of $\subseteq$
$\mathsf{dom}(\Delta_L) \cap \mathsf{dom}(\Delta_R) = \emptyset$	Δ well-formed
$\hat{\alpha}_R \notin \mathsf{dom}(\Delta_L)$	
$\mathsf{dom}(\Gamma) \subseteq \mathsf{dom}(\Delta_L)$	By Definition 5
$\hat{\alpha}_R \notin \mathsf{dom}(\Gamma)$	
$[\Omega_2]\Gamma \vdash [\Omega_2]s : [\Omega_2][\Gamma]A\ \mathbf{!} \gg [\Omega_2]C'\ \not\Downarrow$	Above
Ω_2 and Ω_1 differ only at $\hat{\alpha}$	Above
$\mathsf{FEV}([\Gamma]A) = \emptyset$	Above
$[\Omega_2][\Gamma]A = [\Omega_1][\Gamma]A$	By preceding two lines
$\Gamma \vdash [\Gamma]A\ \mathit{type}$	Above
$\Gamma \longrightarrow \Omega_2$	By Lemma 33 (Extension Transitivity)
$\Omega_2 \vdash [\Gamma]A\ \mathit{type}$	By Lemma 38 (Extension Weakening (Types))
$\mathsf{dom}(\Omega_2) = \mathsf{dom}(\Omega_1)$	Ω_1 and Ω_2 differ only at $\hat{\alpha}$
$\Omega_1 \vdash [\Gamma]A\ \mathit{type}$	By Lemma 18 (Equal Domains)

$\Gamma \vdash [\Gamma]A\ \mathit{type}$ Above
$\Omega \vdash [\Gamma]A\ \mathit{type}$ By Lemma 38 (Extension Weakening (Types))
$[\Omega_1][\Gamma]A = [\Omega'][\Gamma]A = [\Omega][\Gamma]A$ By Lemma 55 (Completing Completeness) (ii) twice
$= [\Omega]A$ By Lemma 29 (Substitution Monotonicity) (iii)

$[\Omega]\Gamma = [\Omega']\Gamma$ By Lemma 57 (Multiple Confluence)
$= [\Omega_1]\Gamma$ By Lemma 57 (Multiple Confluence)
$= [\Omega_2]\Gamma$ Follows from $\hat{\alpha}_R \notin \mathsf{dom}(\Gamma)$

$[\Omega_2]s = [\Omega]s$ Ω_2 and Ω differ only in $\hat{\alpha}$

$[\Omega]\Gamma \vdash [\Omega]s : [\Omega]A\ \mathbf{!} \gg [\Omega_2]C'\ \not{/}\!\!/$ By above equalities

$C = [\Omega']C'$ Above
$[\Omega']C' \neq [\Omega_2]C'$ By def. of subst.
$C \neq [\Omega_2]C'$ By above equality
$C = [\Omega_2]C'$ Instantiating "for all C_2" with $C_2 = [\Omega_2]C'$
$\Rightarrow\Leftarrow$
$\mathsf{FEV}([\Delta]C') = \emptyset$ By contradiction

☞ $\Gamma \vdash s : [\Gamma]A\ \mathbf{!} \gg C'\ \lceil\mathbf{!}\rceil \dashv \Delta$ By SpineRecover

- **Case** $\dfrac{[\Omega]\Gamma \vdash [\Omega]s : [\Omega]A\ p \gg C\ q}{[\Omega]\Gamma \vdash [\Omega]s : [\Omega]A\ p \gg C\ \lceil q\rceil}\ \mathsf{DeclSpinePass}$

$[\Omega]\Gamma \vdash [\Omega]s : [\Omega]A\ p \gg C\ q$ Subderivation
$\Gamma \vdash s : [\Gamma]A\ p \gg C'\ q \dashv \Delta$ By i.h.
☞ $\Delta \longrightarrow \Omega'$ ″
☞ $\mathsf{dom}(\Delta) = \mathsf{dom}(\Omega')$ ″
☞ $\Omega \longrightarrow \Omega'$ ″
☞ $C' = [\Delta]C'$ ″
☞ $C = [\Omega']C'$ ″

We distinguish cases as follows:

  - If $p = \not{/}\!\!/$ or $q = \mathbf{!}$, then we can just apply SpinePass:
    ☞ $\Gamma \vdash s : [\Gamma]A\ p \gg C'\ \lceil q\rceil \dashv \Delta$ By SpinePass
  - Otherwise, $p = \mathbf{!}$ and $q = \not{/}\!\!/$. If $\mathsf{FEV}(C) \neq \emptyset$, we can apply SpinePass, as above. If $\mathsf{FEV}(C) = \emptyset$, then we instead apply SpineRecover:
    ☞ $\Gamma \vdash s : [\Gamma]A\ p \gg C'\ \lceil\mathbf{!}\rceil \dashv \Delta$ By SpineRecover
    Here, $q' = \mathbf{!}$ and $q = \not{/}\!\!/$, so $q' \sqsubseteq q$.

- **Case**
$\dfrac{}{[\Omega]\Gamma \vdash \cdot : [\Omega]A\ p \gg [\Omega]A\ p}\ \mathsf{DeclEmptySpine}$

☞ $\Gamma \vdash \cdot : [\Gamma]A\ p \gg [\Gamma]A\ p \dashv \Gamma$ By EmptySpine
☞ $[\Gamma]A = [\Gamma][\Gamma]A$ By idempotence of substitution
☞ $\Gamma \longrightarrow \Omega$ Given
☞ $\mathsf{dom}(\Gamma) = \mathsf{dom}(\Omega)$ Given
☞ $[\Omega][\Gamma]A = [\Omega]A$ By Lemma 29 (Substitution Monotonicity) (iii)
☞ $\Omega \longrightarrow \Omega$ By Lemma 32 (Extension Reflexivity)

- **Case** $\dfrac{[\Omega]\Gamma \vdash [\Omega]e_0 \Leftarrow [\Omega]A_1\ q \qquad [\Omega]\Gamma \vdash [\Omega]s_0 : [\Omega]A_2\ q \gg B\ p}{[\Omega]\Gamma \vdash [\Omega](e_0\ s_0) : ([\Omega]A_1) \to ([\Omega]A_2)\ q \gg B\ p}\ \mathsf{Decl{\to}Spine}$

	$[\Omega]\Gamma \vdash [\Omega]e_0 \Leftarrow [\Omega]A_1\ q$	Subderivation
	$\Gamma \vdash e_0 \Leftarrow A'\ q \dashv \Theta$	By i.h.
	$\Theta \longrightarrow \Omega_\Theta$	$''$
	$\Omega \longrightarrow \Omega_\Theta$	$''$
	$A = [\Omega_\Theta]A'$	$''$
	$A' = [\Theta]A'$	$''$
	$[\Omega]\Gamma \vdash [\Omega]s_0 : [\Omega]A_2\ q \gg B\ p$	Subderivation
	$\Gamma \vdash s_0 : A_2\ q \gg B\ p \dashv \Delta$	By i.h.
☞	$\Delta \longrightarrow \Omega'$	$''$
☞	$\mathsf{dom}(\Delta) = \mathsf{dom}(\Omega')$	$''$
☞	$\Omega \longrightarrow \Omega'$	$''$
☞	$B' = [\Delta]B'$	$''$
☞	$B = [\Omega']B'$	$''$
☞	$\Gamma \vdash e_0\ s_0 : A_1 \to A_2\ q \gg B\ p \dashv \Delta$	By $\to$Spine

- **Case** $\dfrac{[\Omega]\Gamma \vdash [\Omega]P\ \mathit{true} \qquad [\Omega]\Gamma \vdash [\Omega]e \Leftarrow [\Omega]A_0\ p}{[\Omega]\Gamma \vdash [\Omega]e \Leftarrow ([\Omega]A_0) \wedge [\Omega]P\ p}\ \mathsf{Decl}\wedge\mathsf{I}$

  If e not a case, then:

	$[\Omega]\Gamma \vdash [\Omega]P\ \mathit{true}$	Subderivation
	$\Gamma \vdash P\ \mathit{true} \dashv \Theta$	By Lemma 95 (Completeness of Checkprop)
	$\Theta \longrightarrow \Omega_0'$	$''$
	$\Omega \longrightarrow \Omega_0'$	$''$
	$\Gamma \longrightarrow \Omega$	Given
	$\Gamma \longrightarrow \Omega_0'$	By Lemma 33 (Extension Transitivity)
	$[\Omega]\Gamma = [\Omega]\Omega$	By Lemma 54 (Completing Stability)
	$= [\Omega_0']\Omega_0'$	By Lemma 55 (Completing Completeness) (iii)
	$= [\Omega_0']\Theta$	By Lemma 56 (Confluence of Completeness)
	$\Gamma \vdash A_0 \wedge P\ p\ \mathit{type}$	Given
	$\Gamma \vdash A_0\ p\ \mathit{type}$	By inversion
	$[\Omega]A_0 = [\Omega_0']A_0$	By Lemma 55 (Completing Completeness) (ii)
	$[\Omega]\Gamma \vdash [\Omega]e \Leftarrow [\Omega]A_0\ p$	Subderivation
	$[\Omega_0']\Theta \vdash [\Omega]e \Leftarrow [\Omega_0']A_0\ p$	By above equalities
	$\Theta \vdash e \Leftarrow [\Theta]A_0\ p \dashv \Delta$	By i.h.
☞	$\Delta \longrightarrow \Omega'$	$''$
☞	$\mathsf{dom}(\Delta) = \mathsf{dom}(\Omega')$	$''$
	$\Omega_0' \longrightarrow \Omega'$	$''$
☞	$\Omega \longrightarrow \Omega'$	By Lemma 33 (Extension Transitivity)
☞	$\Gamma \vdash e \Leftarrow A_0 \wedge P\ p \dashv \Delta$	By $\wedge$I

  Otherwise, we have $e = \mathsf{case}(e_0, \Pi)$. Let n be the height of the given derivation.

$n-1$	$[\Omega]\Gamma \vdash [\Omega](\mathsf{case}(e_0, \Pi)) \Leftarrow [\Omega]A_0\ p$	Subderivation
$n-2$	$[\Omega]\Gamma \vdash [\Omega]e_0 \Rightarrow B\ !$	By Lemma 62 (Case Invertibility)
$n-2$	$[\Omega]\Gamma \vdash [\Omega]\Pi :: B \Leftarrow [\Omega]A_0\ p$	$''$
$n-2$	$[\Omega]\Gamma \vdash [\Omega]\Pi\ \mathit{covers}\ B$	$''$
$n-1$	$[\Omega]\Gamma \vdash [\Omega]P\ \mathit{true}$	Subderivation
$n-1$	$[\Omega]\Gamma \vdash [\Omega]\Pi :: B \Leftarrow ([\Omega]A_0) \wedge ([\Omega]P)\ p$	By Lemma 61 (Interpolating With and Exists) (1)
$n-1$	$[\Omega]\Gamma \vdash [\Omega]\Pi :: B \Leftarrow [\Omega](A_0 \wedge P)\ p$	By def. of subst.
	$\Gamma \vdash e_0 \Rightarrow B'\ !\ \dashv \Theta$	By i.h.
	$\Theta \longrightarrow \Omega_0'$	$''$
	$\Omega \longrightarrow \Omega_0'$	$''$
	$B = [\Omega_0']B'$	$''$
	$= [\Omega_0'][\Theta]B'$	By Lemma 30 (Substitution Invariance)
	$[\Omega]\Gamma = [\Omega_0']\Theta$	By Lemma 57 (Multiple Confluence)
	$[\Omega](A_0 \wedge P) = [\Omega_0'](A_0 \wedge P)$	By Lemma 55 (Completing Completeness) (ii)
$n-1$	$[\Omega_0']\Theta \vdash [\Omega]\Pi :: [\Omega_0'][\Theta]B' \Leftarrow [\Omega_0'](A_0 \wedge P)\ p$	By above equalities
	$\Theta \vdash \Pi :: [\Theta]B' \Leftarrow A_0 \wedge P\ p \dashv \Delta$	By i.h.
☞	$\Delta \longrightarrow \Omega'$	$''$
☞	$\mathsf{dom}(\Delta) = \mathsf{dom}(\Omega')$	$''$
	$\Omega_0' \longrightarrow \Omega'$	$''$
	$\Theta \vdash \Pi\ \mathit{covers}\ [\Theta]B'$	By Theorem 10 (Completeness of Match Coverage)
☞	$\Omega \longrightarrow \Omega'$	By Lemma 33 (Extension Transitivity)
☞	$\Gamma \vdash \mathsf{case}(e_0, \Pi) \Leftarrow A_0 \wedge P\ p \dashv \Delta$	By Case

- **Case** DeclNil: Similar to the first part of the Decl$\wedge$I case.

- **Case**
$$\frac{[\Omega]\Gamma \vdash ([\Omega]t) = \mathsf{succ}(t_2)\ \mathit{true} \qquad [\Omega]\Gamma \vdash [\Omega]e_1 \Leftarrow [\Omega]A_0\ p \qquad [\Omega]\Gamma \vdash [\Omega]e_2 \Leftarrow \big(\mathsf{Vec}\ t_2\ [\Omega]A_0\big)\not\!\!\downarrow}{[\Omega]\Gamma \vdash ([\Omega]e_1) :: ([\Omega]e_2) \Leftarrow \big(\mathsf{Vec}\ ([\Omega]t)\ [\Omega]A_0\big)\ p}\ \mathsf{DeclCons}$$

  Let $\Omega^+ = (\Omega, \blacktriangleright_{\hat\alpha}, \hat\alpha : \mathbb{N} = t_2)$.

$$\begin{array}{rll}
 & [\Omega]\Gamma \vdash ([\Omega]t) = \mathsf{succ}(t_2)\ \mathit{true} & \text{Subderivation} \\
 & [\Omega^+](\Gamma, \blacktriangleright_{\hat\alpha}, \hat\alpha : \mathbb{N}) \vdash ([\Omega]t) = [\Omega^+]\mathsf{succ}(\hat\alpha)\ \mathit{true} & \text{Defs. of extension and subst.} \\
1 & \Gamma, \blacktriangleright_{\hat\alpha}, \hat\alpha : \mathbb{N} \vdash t = \mathsf{succ}(\hat\alpha)\ \mathit{true} \dashv \Gamma' & \text{By Lemma 95 (Completeness of Checkprop)} \\
 & \Gamma' \longrightarrow \Omega_0' & '' \\
 & \Omega^+ \longrightarrow \Omega_0' & ''
\end{array}$$

$$\begin{array}{rll}
\Gamma, \blacktriangleright_{\hat\alpha}, \hat\alpha : \mathbb{N} \longrightarrow \Gamma' & & \text{By Lemma 47 (Checkprop Extension)} \\
\Gamma, \blacktriangleright_{\hat\alpha}, \hat\alpha : \mathbb{N} \longrightarrow \Omega_0' & & \text{By Lemma 33 (Extension Transitivity)} \\
[\Omega]\Gamma = [\Omega]\Omega & & \text{By Lemma 54 (Completing Stability)} \\
= [\Omega^+]\Omega^+ & & \text{By def. of context application} \\
= [\Omega_0']\Omega_0' & & \text{By Lemma 55 (Completing Completeness) (iii)} \\
= [\Omega_0']\Gamma' & & \text{By Lemma 56 (Confluence of Completeness)}
\end{array}$$

$$\begin{array}{rll}
[\Omega]A_0 = [\Omega^+]A_0 & & \text{By def. of context application} \\
= [\Omega_0']A_0 & & \text{By Lemma 55 (Completing Completeness) (ii)}
\end{array}$$

$$\begin{array}{rll}
[\Omega]\Gamma \vdash [\Omega]e_1 \Leftarrow [\Omega]A_0\ p & & \text{Subderivation} \\
[\Omega_0']\Gamma' \vdash [\Omega]e_1 \Leftarrow [\Omega_0']A_0\ p & & \text{By above equalities} \\
2 \quad \Gamma' \vdash e_1 \Leftarrow [\Gamma']A_0\ p \dashv \Theta & & \text{By i.h.} \\
\Theta \longrightarrow \Omega_0'' & & '' \\
\Omega_0' \longrightarrow \Omega_0'' & & ''
\end{array}$$

$$\begin{array}{rll}
[\Omega]\Gamma \vdash [\Omega]e_2 \Leftarrow \big(\mathsf{Vec}\ t_2\ [\Omega]A_0\big)\not\!\!\downarrow & & \text{Subderivation} \\
[\Omega]\Gamma \vdash [\Omega]e_2 \Leftarrow \big(\mathsf{Vec}\ ([\Omega^+]\hat\alpha)\ [\Omega]A_0\big)\not\!\!\downarrow & & \text{By def. of substitution} \\
[\Omega_0'']\Theta \vdash [\Omega]e_2 \Leftarrow \big(\mathsf{Vec}\ ([\Omega_0'']\hat\alpha)\ [\Omega_0'']A_0\big)\not\!\!\downarrow & & \text{By lemmas} \\
[\Omega_0'']\Theta \vdash [\Omega]e_2 \Leftarrow [\Omega_0'']\big(\mathsf{Vec}\ \hat\alpha\ A_0\big)\not\!\!\downarrow & & \text{By def. of subst.} \\
3 \quad \Theta \vdash e_2 \Leftarrow [\Theta]A_0\ p \dashv \Delta, \blacktriangleright_{\hat\alpha}, \Delta' & & \text{By i.h.} \\
\Delta, \blacktriangleright_{\hat\alpha}, \Delta' \longrightarrow \Omega'' & & '' \\
\mathsf{dom}(\Delta, \blacktriangleright_{\hat\alpha}, \Delta') = \mathsf{dom}(\Omega'') & & '' \\
\Omega_0'' \longrightarrow \Omega'' & & ''
\end{array}$$

$$\begin{array}{rll}
\Omega'' = (\Omega, \blacktriangleright_{\hat\alpha}, \dots) & & \text{By Lemma 22 (Extension Inversion) (ii)} \\
\text{☞} \quad \Delta \longrightarrow \Omega' & & '' \\
\text{☞} \quad \mathsf{dom}(\Delta) = \mathsf{dom}(\Omega') & & '' \\
(\Gamma', \blacktriangleright_{\hat\alpha}, \dots) \longrightarrow \Omega' & & \text{By Lemma 33 (Extension Transitivity)} \\
\text{☞} \quad \Omega \longrightarrow \Omega' & & \text{By Lemma 22 (Extension Inversion) (ii)}
\end{array}$$

$$\text{☞} \quad \Gamma \vdash e_1 :: e_2 \Leftarrow (\mathsf{Vec}\ t\ A_0)\ p \dashv \Delta \qquad \text{By Cons}$$

- **Case**
$$\frac{[\Omega]\Gamma \vdash [\Omega]e_1 \Leftarrow A_1'\ p \qquad [\Omega]\Gamma \vdash [\Omega]e_2 \Leftarrow A_2'\ p}{[\Omega]\Gamma \vdash \langle [\Omega]e_1, [\Omega]e_2\rangle \Leftarrow A_1' \times A_2'\ p}\ \mathsf{Decl}\times\mathsf{I}$$

  Either $[\Gamma]A = A_1 \times A_2$ or $[\Gamma]A = \hat\alpha \in \mathsf{unsolved}(\Gamma)$.

  – In the first case ($[\Gamma]A = A_1 \times A_2$), we have $A_1' = [\Omega]A_1$ and $A_2' = [\Omega]A_2$.

$[\Omega]\Gamma \vdash [\Omega]e_1 \Leftarrow A_1'\ \mathrm{p}$ — Subderivation
$[\Omega]\Gamma \vdash [\Omega]e_1 \Leftarrow [\Omega]A_1\ \mathrm{p}$ — $[\Omega]A_1 = A_1'$
$\Gamma \vdash e_1 \Leftarrow [\Gamma]A_1\ \mathrm{p} \dashv \Theta$ — By i.h.
$\Theta \longrightarrow \Omega_\Theta$ — ″
$\mathsf{dom}(\Theta) = \mathsf{dom}(\Omega_\Theta)$ — ″
$\Omega \longrightarrow \Omega_\Theta$ — ″

$[\Omega]\Gamma \vdash [\Omega]e_2 \Leftarrow A_2'\ \mathrm{p}$ — Subderivation
$[\Omega]\Gamma \vdash [\Omega]e_2 \Leftarrow [\Omega]A_2\ \mathrm{p}$ — $[\Omega]A_2 = A_2'$

$\Gamma \longrightarrow \Theta$ — By Lemma 51 (Typing Extension)
$[\Omega]\Gamma = [\Omega_\Theta]\Theta$ — By Lemma 57 (Multiple Confluence)
$[\Omega]A_2 = [\Omega_\Theta]A_2$ — By Lemma 55 (Completing Completeness) (ii)

$[\Omega_\Theta]\Theta \vdash [\Omega]e_2 \Leftarrow [\Omega_\Theta]A_2\ \mathrm{p}$ — By above equalities
$\Theta \vdash e_2 \Leftarrow [\Gamma]A_2\ \mathrm{p} \dashv \Delta$ — By i.h.
☞ $\Delta \longrightarrow \Omega'$ — ″
☞ $\mathsf{dom}(\Delta) = \mathsf{dom}(\Omega')$ — ″
$\Omega_\Theta \longrightarrow \Omega'$ — ″
☞ $\Omega \longrightarrow \Omega'$ — By Lemma 33 (Extension Transitivity)
$\Gamma \vdash \langle e_1, e_2\rangle \Leftarrow ([\Gamma]A_1) \times ([\Gamma]A_2)\ \mathrm{p} \dashv \Delta$ — By $\times\mathsf{I}$
☞ $\Gamma \vdash \langle e_1, e_2\rangle \Leftarrow [\Gamma](A_1 \times A_2)\ \mathrm{p} \dashv \Delta$ — By def. of subst.

– In the second case, where $[\Gamma]A = \hat{\alpha}$, combine the corresponding subcase for $\mathsf{Decl}{+}\mathsf{I}_k$ with some straightforward additional reasoning about contexts (because here we have two subderivations, rather than one).

- **Case** $\dfrac{[\Omega]\Gamma \vdash [\Omega]e_0 \Rightarrow \mathrm{C}\ \mathbf{!} \qquad [\Omega]\Gamma \vdash [\Omega]\Pi :: \mathrm{C} \Leftarrow [\Omega]A\ \mathrm{p} \qquad [\Omega]\Gamma \vdash [\Omega]\Pi\ \mathit{covers}\ \mathrm{C}}{[\Omega]\Gamma \vdash \mathsf{case}([\Omega]e_0, [\Omega]\Pi) \Leftarrow [\Omega]A\ \mathrm{p}}\ \mathsf{DeclCase}$

$[\Omega]\Gamma \vdash [\Omega]e_0 \Rightarrow \mathrm{C}\ \mathbf{!}$ — Subderivation
$\Gamma \vdash e_0 \Rightarrow \mathrm{C}'\ \mathbf{!} \dashv \Theta$ — By i.h.
$\Theta \longrightarrow \Omega_\Theta$ — ″
$\mathsf{dom}(\Theta) = \mathsf{dom}(\Omega_\Theta)$ — ″
$\Omega \longrightarrow \Omega_\Theta$ — ″
$\mathrm{C} = [\Omega_\Theta]\mathrm{C}'$ — ″

$\Theta \vdash \mathrm{C}'\ \mathbf{!}\ \mathit{type}$ — By Lemma 63 (Well-Formed Outputs of Typing)
$\mathsf{FEV}(\mathrm{C}') = \emptyset$ — By inversion
$[\Omega_\Theta]\mathrm{C}' = \mathrm{C}'$ — By a property of substitution

	$\Gamma \longrightarrow \Omega$	Given
	$\Delta \longrightarrow \Omega$	Given
	$\Theta \longrightarrow \Omega$	By Lemma 33 (Extension Transitivity)
	$[\Omega]\Gamma = [\Omega]\Theta = [\Omega]\Delta$	By Lemma 56 (Confluence of Completeness)
	$\Gamma \longrightarrow \Theta$	By Lemma 51 (Typing Extension)
	$\Gamma \longrightarrow \Omega_\Theta$	By Lemma 33 (Extension Transitivity)
	$[\Omega]\Gamma = [\Omega_\Theta]\Theta$	By Lemma 57 (Multiple Confluence)
	$\Gamma \vdash A\ \mathit{type}$	Given + inversion
	$\Omega \vdash A\ \mathit{type}$	By Lemma 38 (Extension Weakening (Types))
	$[\Omega]A = [\Omega_\Theta]A$	By Lemma 55 (Completing Completeness) (ii)
	$[\Omega]\Gamma \vdash [\Omega]\Pi :: C \Leftarrow [\Omega]A\ p$	Subderivation
	$[\Omega_\Theta]\Theta \vdash [\Omega]\Pi :: [\Omega_\Theta]C' \Leftarrow [\Omega_\Theta]A\ p$	By above equalities
	$\Theta \vdash \Pi :: C' \Leftarrow [\Theta]A\ p \dashv \Delta$	By i.h. (v)
☞	$\Delta \longrightarrow \Omega'$	$''$
	$\mathsf{dom}(\Delta) = \mathsf{dom}(\Omega')$	$''$
	$\Omega_\Theta \longrightarrow \Omega$	$''$
☞	$\Omega \longrightarrow \Omega'$	By Lemma 33 (Extension Transitivity)

$[\Omega]\Gamma \vdash [\Omega]\Pi\ \mathit{covers}\ C$	Subderivation
$[\Omega]\Gamma = [\Omega]\Delta$	Above
$= [\Omega']\Delta$	By Lemma 57 (Multiple Confluence)
$[\Omega']\Delta \vdash [\Omega]\Pi\ \mathit{covers}\ C'$	By above equalities
$\Delta \longrightarrow \Omega'$	By Lemma 33 (Extension Transitivity)

$\Gamma \vdash C'\ \mathbf{!}\ \mathit{type}$	Given
$\Gamma \longrightarrow \Delta$	By Lemma 51 (Typing Extension) & 33
$\Delta \vdash C'\ \mathbf{!}\ \mathit{type}$	By Lemma 41 (Extension Weakening for Principal Typing)
$[\Delta]C' = C'$	By $\mathsf{FEV}(C') = \emptyset$ and a property of subst.
$\Delta \vdash \Pi\ \mathit{covers}\ C'$	By Theorem 10 (Completeness of Match Coverage)
☞ $\Gamma \vdash \mathsf{case}(e_0, \Pi) \Leftarrow [\Gamma]A\ p \dashv \Delta$	By Case

- **Case** $\dfrac{[\Omega]\Gamma \vdash [\Omega]e_1 \Leftarrow A_1\ p \qquad [\Omega]\Gamma \vdash [\Omega]e_2 \Leftarrow A_2\ p}{[\Omega]\Gamma \vdash \langle [\Omega]e_1, [\Omega]e_2\rangle \Leftarrow \underbrace{A_1 \times A_2}_{[\Omega]A}\ p}\ \mathsf{Decl}\times\mathsf{I}$

  Either $A = \hat{\alpha}$ where $[\Omega]\hat{\alpha} = A_1 \times A_2$, or $A = A_1' \times A_2'$ where $A_1 = [\Omega]A_1'$ and $A_2 = [\Omega]A_2'$.
  In the former case ($A = \hat{\alpha}$):
  We have $[\Omega]\hat{\alpha} = A_1 \times A_2$. Therefore $A_1 = [\Omega]A_1'$ and $A_2 = [\Omega]A_2'$. Moreover, $\Gamma = \Gamma_0[\hat{\alpha} : \kappa]$.

$[\Omega]\Gamma \vdash [\Omega]e_1 \Leftarrow [\Omega]A_1'\ p$	Subderivation
Let $\Gamma' = \Gamma_0[\hat{\alpha}_1 : \kappa, \hat{\alpha}_2 : \kappa, \hat{\alpha} : \kappa = \hat{\alpha}_1 + \hat{\alpha}_2]$.	
$[\Omega]\Gamma = [\Omega]\Gamma'$	By def. of context substitution
$[\Omega]\Gamma' \vdash [\Omega]e_1 \Leftarrow [\Omega]A_1'\ p$	By above equality
$\Gamma' \vdash e_1 \Leftarrow [\Gamma']A_1'\ p' \dashv \Theta$	By i.h.
$\Theta \longrightarrow \Omega_1$	$''$
$\Omega \longrightarrow \Omega_1$	$''$
$\mathsf{dom}(\Theta) = \mathsf{dom}(\Omega_1)$	$''$
$[\Omega]\Gamma \vdash [\Omega]e_2 \Leftarrow [\Omega]A_2'\ p$	Subderivation

	$[\Omega]\Gamma = [\Omega_1]\Theta$	By Lemma 57 (Multiple Confluence)
	$[\Omega]A_2' = [\Omega_1]A_2'$	By Lemma 55 (Completing Completeness) (ii)
	$[\Omega_1]\Theta \vdash [\Omega]e_2 \Leftarrow [\Omega_1]A_2'\ \mathrm{p}$	By above equalities
	$\Theta \vdash e_2 \Leftarrow [\Theta]A_2'\ \mathrm{p}' \dashv \Delta$	By i.h.
☞	$\mathsf{dom}(\Delta) = \mathsf{dom}(\Omega')$	$''$
☞	$\Delta \longrightarrow \Omega'$	$''$
	$\Omega_1 \longrightarrow \Omega'$	$''$
☞	$\Omega \longrightarrow \Omega'$	By Lemma 33 (Extension Transitivity)
☞	$\Gamma \vdash \langle e_1, e_2\rangle \Leftarrow \hat{\alpha}\ \mathrm{p}' \dashv \Delta$	By $\times\mathsf{I}\hat{\alpha}$

In the latter case ($A = A_1' \times A_2'$):

	$[\Omega]\Gamma \vdash [\Omega]e_1 \Leftarrow A_1\ \mathrm{p}$	Subderivation
	$[\Omega]\Gamma \vdash [\Omega]e_1 \Leftarrow [\Omega]A_1'\ \mathrm{p}$	$A_1 = [\Omega]A_1'$
	$\Gamma \vdash e_1 \Leftarrow [\Gamma]A_1'\ \mathrm{p} \dashv \Theta$	By i.h.
	$\Theta \longrightarrow \Omega_0$	$''$
	$\mathsf{dom}(\Theta) = \mathsf{dom}(\Omega_0)$	$''$
	$\Omega \longrightarrow \Omega_0$	$''$
	$[\Omega]\Gamma \vdash [\Omega]e_2 \Leftarrow A_2\ \mathrm{p}$	Subderivation
	$[\Omega]\Gamma \vdash [\Omega]e_2 \Leftarrow [\Omega]A_2'\ \mathrm{p}$	$A_2 = [\Omega]A_2'$
	$\Gamma \vdash A_1' \times A_2'\ \mathrm{p}\ \mathit{type}$	Given ($A = A_1' \times A_2'$)
	$\Gamma \vdash A_2'\ \mathit{type}$	By inversion
	$\Gamma \longrightarrow \Omega$	Given
	$\Gamma \longrightarrow \Omega_0$	By Lemma 33 (Extension Transitivity)
	$\Omega_0 \vdash A_2'\ \mathit{type}$	By Lemma 38 (Extension Weakening (Types))
	$[\Omega]\Gamma \vdash [\Omega]e_2 \Leftarrow [\Omega_0]A_2'\ \mathrm{p}$	By Lemma 55 (Completing Completeness)
	$[\Omega]\Gamma \vdash [\Omega]e_2 \Leftarrow [\Omega_0][\Theta]A_2'\ \mathrm{p}$	By Lemma 29 (Substitution Monotonicity) (iii)
	$[\Omega]\Theta \vdash [\Omega]e_2 \Leftarrow [\Omega_0][\Theta]A_2'\ \mathrm{p}$	By Lemma 57 (Multiple Confluence)
	$\Theta \vdash e_2 \Leftarrow [\Theta]A_2'\ \mathrm{p} \dashv \Delta$	By i.h.
☞	$\Delta \longrightarrow \Omega'$	$''$
☞	$\mathsf{dom}(\Delta) = \mathsf{dom}(\Omega')$	$''$
	$\Omega_0 \longrightarrow \Omega'$	$''$
☞	$\Omega \longrightarrow \Omega'$	By Lemma 33 (Extension Transitivity)
	$\Gamma \vdash \langle e_1, e_2\rangle \Leftarrow ([\Omega]A_1) \times ([\Omega]A_2)\ \mathrm{p} \dashv \Delta$	By $\times\mathsf{I}$
☞	$\Gamma \vdash \langle e_1, e_2\rangle \Leftarrow [\Omega](A_1 \times A_2)\ \mathrm{p} \dashv \Delta$	By def. of substitution

Now we turn to parts (v) and (vi), completeness of matching.

- **Case** DeclMatchEmpty: Apply rule MatchEmpty.
- **Case** DeclMatchSeq: Apply the i.h. twice, along with standard lemmas.
- **Case** DeclMatchBase: Apply the i.h. (i) and rule MatchBase.
- **Case** DeclMatchUnit: Apply the i.h. and rule MatchUnit.
- **Case** DeclMatch$\exists$: By i.h. and rule Match$\exists$.
- **Case** DeclMatch$\times$: By i.h. and rule Match$\times$.
- **Case** DeclMatch$+_k$: By i.h. and rule Match$+_k$.
- **Case**
$$\frac{[\Omega]\Gamma\ /\ \mathrm{P} \vdash \vec{\rho} \Rightarrow e :: [\Omega]A, [\Omega]\vec{A} \Leftarrow [\Omega]\mathrm{C}\ \mathrm{p}}{[\Omega]\Gamma \vdash \vec{\rho} \Rightarrow e :: ([\Omega]A \wedge [\Omega]\mathrm{P}), [\Omega]\vec{A} \Leftarrow [\Omega]\mathrm{C}\ \mathrm{p}}\ \mathsf{DeclMatch}\wedge$$
To apply the i.h. (vi), we will show (1) $\Gamma \vdash (A, \vec{A})$ **!** *types*, (2) $\Gamma \vdash \mathrm{P}$ *prop*, (3) $\mathsf{FEV}(\mathrm{P}) = \emptyset$, (4) $\Gamma \vdash \mathrm{C}\ \mathrm{p}$ *type*, (5) $[\Omega]\Gamma\ /\ [\Omega]\mathrm{P} \vdash \vec{\rho} \Rightarrow [\Omega]e :: [\Omega]\vec{A} \Leftarrow [\Omega]\mathrm{C}\ \mathrm{p}$, and (6) $\mathrm{p}' \sqsubseteq \mathrm{p}$.

$\Gamma \vdash (A \wedge P, \vec{A})$ **!** *types* Given
$\Gamma \vdash (A \wedge P)$ **!** *type* By inversion on PrincipalTypevecWF
$\Gamma \vdash A$ **!** *type* By Lemma 42 (Inversion of Principal Typing) (3)
(2) $\Gamma \vdash P$ *prop* ″
(3) $\mathsf{FEV}(P) = \emptyset$ By inversion
(1) $\Gamma \vdash (A, \vec{A})$ **!** *types* By inversion and PrincipalTypevecWF
(4) $\Gamma \vdash C\ p$ *type* Given
(5) $[\Omega]\Gamma \mathbin{/} P \vdash \vec{\rho} \Rightarrow [\Omega]e :: [\Omega]A, [\Omega]\vec{A} \Leftarrow [\Omega]C\ p$ Subderivation
(6) $p' \sqsubseteq p$ Given

$\Gamma \mathbin{/} [\Gamma]P \vdash \vec{\rho} \Rightarrow e :: [\Gamma](A, \vec{A}) \Leftarrow [\Gamma]C\ p' \dashv \Delta$ By i.h. (vi)
☞ $\Delta \longrightarrow \Omega'$ ″
☞ $\mathsf{dom}(\Delta) = \mathsf{dom}(\Omega')$ ″
☞ $\Omega \longrightarrow \Omega'$ ″

$\Gamma \mathbin{/} [\Gamma]P \vdash \vec{\rho} \Rightarrow e :: [\Gamma]A, [\Gamma]\vec{A}) \Leftarrow [\Gamma]C\ p' \dashv \Delta$ By def. of subst.
$\Gamma \vdash \vec{\rho} \Rightarrow e :: ([\Gamma]A \wedge [\Gamma]P), [\Gamma]\vec{A}) \Leftarrow [\Gamma]C\ p' \dashv \Delta$ By Match$\wedge$
☞ $\Gamma \vdash \vec{\rho} \Rightarrow e :: [\Gamma]\big((A \wedge P), \vec{A}\big) \Leftarrow [\Gamma]C\ p' \dashv \Delta$ By def. of subst.

- **Case** DeclMatchNeg: By i.h. and rule MatchNeg.
- **Case** DeclMatchWild: By i.h. and rule MatchWild.
- **Case** DeclMatchNil: Similar to the DeclMatch$\wedge$ case.
- **Case** DeclMatchCons: Similar to the DeclMatch$\exists$ and DeclMatch$\wedge$ cases.
- **Case**
$$\frac{\mathsf{mgu}([\Omega]\sigma, [\Omega]\tau) = \bot}{[\Omega]\Gamma \mathbin{/} [\Omega]\sigma = [\Omega]\tau \vdash [\Omega](\vec{\rho} \Rightarrow e) :: [\Omega]\vec{A} \Leftarrow [\Omega]C\ p}\ \mathsf{DeclMatch}\bot$$

☞ $\Gamma \longrightarrow \Omega$ Given
$\mathsf{FEV}(\sigma = \tau) = \emptyset$ Given
$[\Omega]\sigma = [\Gamma]\sigma$ By Lemma 39 (Principal Agreement) (i)
$[\Omega]\tau = [\Gamma]\tau$ Similar

$\mathsf{mgu}([\Omega]\sigma, [\Omega]\tau) = \bot$ Given
$\mathsf{mgu}([\Gamma]\sigma, [\Gamma]\tau) = \bot$ By above equalities
$\Gamma \mathbin{/} \sigma \doteq \tau : \kappa \dashv \bot$ By Lemma 92 (Completeness of Elimeq) (2)
☞ $\Gamma \mathbin{/} [\Gamma]\sigma = [\Gamma]\tau \vdash \vec{\rho} \Rightarrow e :: [\Gamma]\vec{A} \Leftarrow [\Gamma]C\ p \dashv \Gamma$ By Match$\bot$

☞ $\Omega \longrightarrow \Omega$ By Lemma 32 (Extension Reflexivity)
☞ $\mathsf{dom}(\Gamma) = \mathsf{dom}(\Omega)$ Given

- **Case**
$$\frac{\mathsf{mgu}([\Omega]\sigma, [\Omega]\tau) = \theta \qquad \theta([\Omega]\Gamma) \vdash \theta(\vec{\rho} \Rightarrow [\Omega]e) :: \theta([\Omega]\vec{A}) \Leftarrow \theta([\Omega]C)\ p}{[\Omega]\Gamma \mathbin{/} [\Omega]\sigma = [\Omega]\tau \vdash \vec{\rho} \Rightarrow [\Omega]e :: [\Omega]\vec{A} \Leftarrow [\Omega]C\ p}\ \mathsf{DeclMatchUnify}$$

$([\Omega]\sigma = [\Gamma]\sigma)$ and $([\Omega]\tau = [\Gamma]\tau)$ As in DeclMatch$\bot$ case
$\mathsf{mgu}([\Omega]\sigma, [\Omega]\tau) = \theta$ Given
$\mathsf{mgu}([\Gamma]\sigma, [\Gamma]\tau) = \theta$ By above equalities

$\Gamma \mathbin{/} \sigma \doteq \tau : \kappa \dashv (\Gamma, \Theta)$ By Lemma 92 (Completeness of Elimeq) (1)
$\Theta = (\alpha_1 = t_1, \ldots, \alpha_n = t_n)$ ″
$[\Gamma, \Theta]u = \theta([\Gamma]u)$ ″ for all $\Gamma \vdash u : \kappa$

$\theta([\Omega]\Gamma) \vdash \theta(\vec{\rho} \Rightarrow [\Omega]e) :: \theta([\Omega]\vec{A}) \Leftarrow \theta([\Omega]C)\ p$ Subderivation

$\theta([\Omega]\Gamma) = [\Omega, \blacktriangleright_p, \Theta](\Gamma, \blacktriangleright_p, \Theta)$ By Lemma 93 (Substitution Upgrade) (iii)
$\theta([\Omega]\vec{A}) = [\Omega, \blacktriangleright_p, \Theta]\vec{A}$ By Lemma 93 (Substitution Upgrade) (i) (over $\vec{A}$)
$\theta([\Omega]C) = [\Omega, \blacktriangleright_p, \Theta]C$ By Lemma 93 (Substitution Upgrade) (i)
$\theta(\vec{\rho} \Rightarrow [\Omega]e) = [\Omega, \blacktriangleright_p, \Theta](\vec{\rho} \Rightarrow e)$ By Lemma 93 (Substitution Upgrade) (iv)

$[\Omega, \blacktriangleright_p, \Theta](\Gamma, \blacktriangleright_p, \Theta) \vdash [\Omega, \blacktriangleright_p, \Theta](\vec{\rho} \Rightarrow e) :: [\Omega, \blacktriangleright_p, \Theta]\vec{A} \Leftarrow [\Omega, \blacktriangleright_p, \Theta]C\ p$ By above equalities

$\Gamma, \blacktriangleright_p, \Theta \vdash (\vec{\rho} \Rightarrow e) :: [\Gamma, \blacktriangleright_p, \Theta]\vec{A} \Leftarrow [\Gamma, \blacktriangleright_p, \Theta]C\ p \dashv \Delta, \blacktriangleright_p, \Delta'$ By i.h.
$\Delta, \blacktriangleright_p, \Delta' \longrightarrow \Omega', \blacktriangleright_p, \Omega''$ ″
$\Omega, \blacktriangleright_p, \Theta \longrightarrow \Omega', \blacktriangleright_p, \Omega''$ ″
$\mathsf{dom}(\Delta, \blacktriangleright_p, \Delta') = \mathsf{dom}(\Omega', \blacktriangleright_p, \Omega'')$ ″

☞ $\Delta \longrightarrow \Omega'$ By Lemma 22 (Extension Inversion) (ii)
☞ $\mathsf{dom}(\Delta) = \mathsf{dom}(\Omega')$ ″
☞ $\Omega \longrightarrow \Omega'$ By Lemma 22 (Extension Inversion) (ii)

☞ $\Gamma\ /\ [\Gamma]\sigma = [\Gamma]\tau \vdash \vec{\rho} \Rightarrow e :: [\Gamma]\vec{A} \Leftarrow [\Gamma]C\ p \dashv \Delta$ By MatchUnify □